

МІНІСТЕРСТВО ОСВІТИ Й НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
“Розробка програмного забезпечення для мобільних пристроїв”

для студентів спеціальності
“Інженерія програмного забезпечення”

2016

Методичні вказівки до виконання лабораторних робіт з дисципліни “Розробка програмного забезпечення для мобільних пристроїв ” для студентів спеціальності “Інженерія програмного забезпечення”/ Укл.: О.О. Степаненко, Л.П. Скачко.– Запоріжжя: ЗНТУ, 2016. – 58 с.

Автори: Степаненко О.О., к.т.зв., доцент;
Скачко Л.П., асистент

Рецензент: Онищенко В.Ф., к. ф.-м. н., доцент

Відповідальний за випуск: Субботін С.О.

Затверджено
вченою радою інституту
інформатики та
радіоелектроніки

Протокол № 1
від “19” березня 2014 р.

Затверджено
на засіданні кафедри
“Програмні засоби”

Протокол № 7
від “06” березня 2014 р.

ЗМІСТ

Вступ	5
Лабораторна робота № 1 Структура мобільних телефонів. Що таке Java? Історія створення.	6
Лабораторна робота № 2 Засоби розробки мобільних застосунків. Середовище програмування SUN ONE Studio 4 ME. Телефонні емулятори.	12
Лабораторна робота № 3 Механізм роботи додатків Java 2 ME. Мідлети. Створення інтерфейсу користувача.	19
Лабораторна робота № 4 Використання графіки в мобільних телефонах. Технологія створення ігор.	28
Висновки	35
Література	36

Людські потреби бути завжди в курсі справ виводять інформаційні технології на пряму лінію по створенню всіх нових пристроїв і гаджетів. Незручність експлуатації комп'ютерів і ноутбуків спричиняє появу різних міні-комп'ютерів, смартфонів та комунікаторів, в основі яких лежить усе та ж операційна система. Лідуючі позиції на сьогоднішній день займають платформи Android і iPhone. Але ці платформи можуть працювати повноцінно тільки при одній умові - якщо для них буде здійснюватись розробка мобільних додатків.

Метою курсу “Програмування малих та мобільних платформ” є набуття навиків створення сучасних мобільних додатків високої якості, знайомство з сучасними середами та мовами розробок програмного забезпечення для мобільних пристроїв, вміння правильно та ефективно тестувати розроблені мобільні додатки.

Звіт з лабораторної роботи повинен містити:

- тему;
- мету;
- текстовий та графічний матеріал розробки;
- висновки.

Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання по теоретичному матеріалу, про методи виконання кожного етапу роботи, по змісту основних розділів оформленого звіту з демонстрацією результатів на конкретних прикладах. Студент повинен вміти правильно аналізувати отримані результати. Для самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

Вступ

Ринок мобільних телефонів - це, мабуть, самий стрімкозростаючий сегмент ринку. Десять років тому мобільний телефон був чимось дорогим і недосяжним. Оплата за зв'язок була астрономічною. Але часи міняються, зв'язок подешевшав, мобільні телефони продаються на кожному розі і наявність телефону у людини можна вже порівняти з наявністю звичайного наручного годинника.

Велика кількість різноманітних моделей телефонів створює на цьому ринку велику конкуренцію. Безліч виробників прагнуть включити в телефони якомога більше різних функцій. Присутністю в телефоні адресної книги, органайзера, будильника і навіть лічильника калорій вже нікого не здивуєш, і в якийсь момент часу виникло питання, а чим же ще можна обладнати телефон? Найідеальнішим рішенням цього питання могла стати можливість завантаження в телефон програм сторонніх виробників. Не секрет, що основна маса телефонів працює на основі прошивки, тому завантаження програм в телефон не можливе. Аби така можливість з'явилася, виробникам треба було б вирішити низку складних запитань. По-перше - це мова програмування, на якому писатимуться програми для телефонів і, відповідно, підтримка цієї мови самим телефоном. Друге - це апаратна архітектура мобільних телефонів, яка сильно обмежена в своїх системних ресурсах, а це процесор, пам'ять, графічний інтерфейс. Поліпшення архітектури незмінно веде до збільшення вартості телефону, тому ідеальної в цьому випадку стала платформа Java 2 Micro Edition, створена компанією Sun Microsystems.

Свою роботу я почав з історії появи і розвитку Java. Це дозволить краще зрозуміти особливості платформи і спектр існуючих продуктів і технологій. Також створення Java є цікавим прикладом історії одного з найпопулярніших і успішніших проектів на комп'ютерному світі. Потім викладаються основні концепції ООП, необхідні для освоєння об'єктно-орієнтованої мови програмування Java. Далі розглянеться платформа Java 2 Micro Edition, що є стандартом в програмуванні додатків для мобільних телефонів.

Метою роботи є таке завдання: вивчення основних конфігурацій і профілів платформи Java 2 Micro Edition, розглянути дві версії профілів MIDP 1.0 і MIDP 2.0, що є на даний момент, на базі яких відбувається побудова мобільних застосувань. Вивчити основи побудови мобільних програм, використання графіки, техніка створення ігор для телефонів, робота із звуком. Предметом дослідження є мова програмування Java та платформа Java 2 Micro Edition.

ЛАБОРАТОРНА РОБОТА №1.

Структура мобільних телефонів. Що таке Java? Історія створення.

Мета: зайомство з мобільними пристроями, їх апаратною та програмною структурою та з популярною мовою розробки мобільного додатку Java.

Короткі теоретичні відомості

Перш ніж приступити до вивчення особливостей програмування мобільних телефонів на Java 2 ME, необхідно познайомитися із загальною схемою пристрою телефону.

Зараз на ринку мобільних телефонів існує безліч як дрібних, так і крупних виробників мобільних телефонів, що пропонують широкий асортимент продукції. За ціною і по своїх функціональних ознаках наявні пристрої, можна умовно розділити на три категорії. Перша є мобільними телефонами з набором простих функцій, такі телефони часто називають «звонилка». Функціональність подібних телефонів зводиться лише до їх прямого призначення. З такого пристрою можна виробити або прийняти виклик і скористатися послугами SMS. Друга категорія мобільних телефонів має ширший круг можливостей. Дисплей таких телефонів зазвичай кольоровий, розмір варіюється від 80 до 130 пікселів з глибиною кольору близько п'яти тисяч кольорів. Телефони цієї категорії зазвичай містять ті ж можливості, що і в апаратах першої категорії і ряд додаткових функцій. Одна них - це можливість завантаження додатків написаних на мові Java. Третя категорія телефонів є так званою бізнесом-категорією. Подібні пристрої мають вражаючі набори функцій, порівнянні з кишеньковими портативними пристроями (КПК), а ціна таких мобільних телефонів може бути достатня високою. Все три умовні категорії телефонів схожі одним - функцією зв'язку, а відмінність в додаткових можливостях викликана умовами конкуренції між виробниками і необхідністю просування нових технологій. Технічний прогрес не стоїть на місці, створюються всі менші по розмірах процесори з потужнішими характеристиками. Як правило, всі телефони для зв'язку використовують стандарт GSM.

Стандарт GSM

Цифровий стандарт GSM (Global System Mobile Communication - система мобільного глобального зв'язку), на основі якого відбувається зв'язок між телефонами, - найпоширеніший стандарт у всьому світі, що забезпечує діапазон роботи мобільних телефонів на частотах 900/1800/1900 Мгц. Стандарт GSM володіє якісними і безпечними характеристиками зв'язку, що і вплинуло на його популярність у всьому світі. У нашій країні це основний стандарт, що використовується всіма крупними операторами зв'язку.

Архітектура телефонів

Телефон має процесор, пам'ять, прийомо-передатчик, систему контролерів для дисплея, клавіатури, акумулятора і карту SIM (Subscriber identification module - ідентифікаційний модуль абонента). Процесор разом з пам'яттю є серцем телефону. Він має скромніші характеристики, ніж процесор комп'ютера. Прийомо-передатчик служить для прийому і передачі сигналу і включає різні фільтри, кодери, декодери мови і сигналів. Система контролерів для клавіатури (клавіш телефону) і дисплея дозволяє вводити інформацію в телефон за допомогою клавіш і виводити її на дисплей. Контролер акумулятора виробляє управління процесом зарядки акумуляторних батарей телефону.

Все апаратне забезпечення телефону (залізо) безумовне, само по собі функціонувати не може, і робота здійснюється на основі прошивки або операційної системи.

Прошивка і операційна система

Все наявні мобільні телефони можна розділити ще на двох типів - це телефони, що працюють під управлінням прошивки і телефони, що використовують операційну систему. Телефони з операційною системою також носять назву смартфони. На ринку зараз є дві операційні системи, адаптовані під потреби телефонів, це Windows Mobile і Symbian.

Прошивка телефону - це програма, що управляє роботою телефону, але ушита в сам телефон. Кожен з виробників створює свої прошивки для конкретних моделей, зумовлюючи тим самим функціональність телефону. Операційна система дозволяє здійснювати завантаження створених програм в телефон, а в прошивці така можливість не передбачена. Точніше сказати, не була передбачена до появи платформи Java 2 ME. Для роботи додатків написаних на Java, необхідна віртуальна машина або середовище

виконання додатків Java. Віртуальна машина Java, адаптована спеціально для телефонів, настільки компактна за розміром, і, що саме головне, безпечна, що практично всі виробники телефонів прагнуть створити підтримку Java програм, тобто вбудувати середовище виконання Java програм в телефон. Завантаження сторонніх застосувань дало дуже потужний імпульс мобільної ігрової індустрії, що, як мені здається, вивело телефон на новий якісніший рівень. Єдине що необхідно розуміти в цьому аспекті, це те, що середовище виконання Java-додатків в телефоні реалізоване таким чином, що не дозволяє написати «шкідливі» програми що псують апаратну частину телефону. Модель роботи Java-програм реалізована як би у вигляді закупореної бочки, в якій і відбувається робота всіх Java-додатків. З цієї бочки програми вибратися не можуть, а відповідно не можуть і завдати збитку телефону.

Завантаження Java програм в телефон відбувається за допомогою програмного забезпечення і кабелю, що сполучає телефон з комп'ютером. Також можна завантажити в телефон програми за допомогою інфрачервоного порту, технології Bluetooth, або безпосередньо з Інтернету за допомогою протоколів WAP і GPRS. Всі програми, що завантажуються в телефон, поширюються у вигляді двох файлів, JAD це дескриптор додатка і JAR - архів програми.

Інфрачервоний порт

Інфрачервоний порт мають не всі телефони. В основному це моделі, що відносяться до другої і третьої умовної цінової категорії. За допомогою інфрачервоного порту можна виробити з'єднання з *комп'ютером*, іншим телефоном або будь-яким іншим пристроєм що має інфрачервоний порт. Принцип роботи інфрачервоного порту дуже простий: є світлодіод і фотодіод, відповідно для передачі і прийому сигналу. Помістивши два пристрої на відстані не більше півтора метрів в зоні прямої видимості можна здійснити зв'язок між цими пристроями. Що стосується зв'язку телефону з комп'ютером за допомогою інфрачервоного порту, то це відбувається за допомогою програмного забезпечення що поставляється виробниками телефонів. Зокрема на компакт-диск можна знайти програми для телефонів Sony Ericsson, Nokia і Siemens, детальний аналіз цих програм ви знайдете в главі

Технологія Bluetooth

За допомогою технології Bluetooth можна здійснювати з'єднання відразу з безліччю пристроїв і організовувати невеликі локальні мережі. Зв'язок з використанням

Bluetooth відбувається на частоті 2,4 ГГц з відстані до 10 метрів і що найголовніше, не вимагає прямого з'єднання, тобто пристрої можуть знаходитися поза зоною видимості один одного. Приймач сигналу виконується у вигляді мікросхеми і може зв'язати в мережу близько десятка всіляких пристроїв. Зараз технологія Bluetooth дуже активно використовується виробниками мобільних телефонів для створення безпроводних гарнітур, наприклад навушника і мікрофону.

Протокол WAP

Протокол WAP (Wireless Application Protocol) - це протокол безпроводного з'єднання з мережею Інтернет. Маючи підтримку WAP в мобільному телефоні, з'являється можливість виходу в Інтернет для перегляду спеціально адаптованих web-сторінок на телефоні. Також за допомогою WAP можна викачати або купити гру або додаток, написаний на мові Java. Вартість таких програм зазвичай не перевищує 1-2 долари. Проте користуватися WAP можна лише в тому випадку, якщо ваш оператор стільникового зв'язку надає WAP послуги. Телефон, що має підтримку WAP, забезпечений і WAP-браузером, що і робить можливим перегляд WAP сторінок в Інтернеті. Єдина проблема WAP-сервісу - це висока ціна з'єднання, оскільки оплата відбувається щохвилини, а платите ви практично як за хвилину розмови по телефону. Це виходить вельми дороге задоволення. Частково цю проблему вирішує протокол GPRS.

Протокол GPRS

Зв'язок телефону по *протоколу GPRS* (General Packet Radio Service - радіо сервіс пакетної передачі даних) здійснюється в куди більш щадному режимі. Використовуючи з'єднання WAP, ви фактично дзвоните по іншому номеру, займаючи тим самим один з багаточисельних каналів зв'язку, тому і оплата стягується щохвилини. З використанням GPRS оплата відбувається лише за кількість викачаних даних, вимірювану в байтах, кілобайтах або мегабайтах. Краще використовувати WAP з'єднання, тобто переглядати WAP-сайти, через протокол GPRS, значно економлячи засоби. Все це відбувається завдяки тому, що GPRS використовує пакетну передачу даних і не застосовує якийсь конкретний канал зв'язку, він уклинюється у вільні зони різних каналів, передаючи інформацію у вигляді невеликих пакетів. Зв'язок через GPRS можна використовувати і на комп'ютері, для виходу в Інтернет через мобільний телефон. Така послуга зв'язку носить назву мобільний Інтернет.

SIM Карта

Після придбання телефону необхідно поклопотатися про покупку *SIM-карти* одного з операторів мобільного зв'язку. Купивши SIM-карту, ви стаєте повноцінним абонентом вибраного вами оператора зв'язку. Сама SIM-карта виконана у вигляді невеликого по розмірах пластмасового прямокутника, з контактною смугою на поверхні для стиковки з ідентичним роз'ємом на телефоні. Кожна SIM-карта, містить свій унікальний ідентифікаційний номер, що зрештою вирішує проблему однакових номерів. У SIM-карті також знаходиться простий процесор, і ви можете зберігати на SIM-карте записи, як правило, це звичайна телефонна книга, приблизно на 200-300 записів.

Таким чином, мобільний телефон віддалено нагадує пристрій комп'ютерних систем. Тому адаптація платформи Java 2 ME, для телефонів, сталася абсолютно безболісно, переклавши виток розвитку мобільної індустрії на новий, до теперішнього часу не ведений рівень. Наступна глава розкриває теорію будови платформи Java 2 Micro Edition.

Що таке Java? Історія створення

Java - новітня об'єктно-орієнтована мова, легка у вивченні і дає можливість створювати програми, які можуть виконуватися на будь-якій платформі без яких-небудь доопрацювань (кроссплатформенність).



Мова програмування Java зародилася в 1991 р. в лабораторіях компанії Sun Microsystems. Розробку проекту започаткував Джеймс Гослінг. Створення першої робочої версії, яка мала назву «Oak», зайняло 18 місяців; у 1995 р. проект було перейменовано на «Java», де назву було взято зі списку випадкових слів. Головним мотивом створення Java була потреба мови програмування, яка б не залежала від платформи (тобто від архітектури) і яку можна було б використовувати для створення програмного забезпечення, яке вбудовується в різноманітні побутові електронні прилади, такі як мобільні засоби зв'язку, пристрої дистанційного керування тощо. Досить скоро майже всі найпопулярніші тогочасні

веб-браузери отримали можливість запускати «безпечні» для системи Java аплети всередині веб-сторінок. У грудні 1998 р. Sun Microsystems випустила Java 2 (спершу під назвою J2SE 1.2), де було реалізовано декілька конфігурацій для різних типів платформ. Наприклад, J2EE призначалася для створення корпоративних додатків, а значно урізана J2ME для приладів з обмеженими ресурсами, таких як мобільні телефони. У 2006 році в маркетингових цілях, Версії J2 було перейменовано у Java EE, Java ME та Java SE, відповідно.

13 листопада 2006 року Sun випустили більшу частину Java в якості вільного та відкритого програмного забезпечення у відповідності з умовами GNU General Public License (GPL). 8 травня 2007 корпорація закінчила процес, в результаті якого всі початкові коди Java були випущені під GPL, за винятком невеликої частини коду, на який Sun не мала авторського права.

Завдання на роботу

Ознайомитися з структурою мобільного пристрою, його апаратною та програмною частинами, з історією виникнення мови програмування Java.

Зміст звіту

– Оформляти звіт не потрібно.

Контрольні питання

1. Які основні частини мобільного пристрою ви знаєте?
2. В чому полягає відмінність між операційною системою та прошивкою мобільного пристрою?
3. Які засоби комунікації та передачі інформації в телефоні ви знаєте?
4. Які протоколи обміну інформацією існують?
5. Що таке Java? Коли і як вона виникла?
6. Дайте визначення мобільному пристрою.
7. На яких частотах працює стандарт GSM?

ЛАБОРАТОРНА РОБОТА №2.

Засоби розробки мобільних застосунків. Середовище програмування SUN ONE Studio 4 Mobile Edition. Телефонні емулятори.

Мета: ознайомитися з засобами розробки застосунків та середовищем розробки мобільного додатку SUN ONE Studio 4 Mobile Edition. Навитися розгортати та налагоджувати це середовище.

Короткі теоретичні відомості

Засоби розробки мобільних застосунків

У попередній роботі була описана загальна архітектура платформи Java 2 ME, в цій - вашій увазі будуть представлено два інтегровані середовища для розробки мобільних застосунків - це SUN ONE Studio 4 Mobile Edition та J2ME Wireless Toolkit 2.1. Обидва середовища поширюється безкоштовно і розроблені в надрах компанії Sun Microsystems. Ці інструментарії, на відміну від інших середовищ програмування, адаптовані саме для створення додатків для мобільних пристроїв.

Перш ніж приступити до інсталяції SUN ONE Studio 4 Mobile Edition і J2ME Wireless Toolkit 2.1, необхідно встановити набір інструментальних засобів Java 2 SDK, SE v1.4.2_03 (Software Development Toolkit Standard Edition). Це остання на даний момент версія бібліотеки, що надає, програмні засоби для створення додатків на Java. До складу цієї бібліотеки також входить Java 2 Runtime Environment, SE v1.4.2_03, яка необхідна для роботи додатків написаних на Java.

Установка Java 2 SDK SE

Подвійне клацання лівої кнопки мишки на файлі j2sdk-1_4_2_03-windows-i586-p з каталогу \SDK, що знаходиться на компакт-диск диску, приведе до розпаковування архіву. Далі з'явиться вікно з ліцензійною угодою, натискує на кнопку Next, ви попадете у вікно Java 2 SDK SE v1.4.2_03 - Custom Setup.

У цьому діалоговому вікні вам буде запропоновано вибрати каталог і типа установки. Нам потрібні всі компоненти, тому натискуйте кнопку Next. У наступному вікні вам необхідно вибрати браузер для інтеграції Java, зупините свій вибір на необхідному і підтвердите свої дії, після чого почнеться процес установки, в кінці якого

вам запропонують перезавантажити комп'ютер. Після установки Java 2 SDK, можна приступати до інсталяції SUN ONE Studio 4 Mobile Edition і J2ME Wireless Toolkit.

Відразу хотілося б зупинитися на деяких особливостях тих, що виникають при інсталяції. Обоє середовища програмування незалежні один від одного, але в складі, а точніше в комплекті з SUN ONE Studio 4 Mobile Edition, входить J2ME Wireless Toolkit версії 1.0.4 01, але на даний момент вже є J2ME Wireless Toolkit версії 2.1. Така плутанина обумовлена якраз тим, що обоє продукту незалежні один від одного і розвиваються паралельно. Розгля немо установку потужнішого середовища SUN ONE Studio 4 Mobile Edition.

Середовище програмування SUN ONE Studio 4 Mobile Edition

На компакт-диску в директорії \SDK знаходиться файл ffj_nic_win32 - це програма установки інтегрованого середовища розробки SUN ONE Studio 4 Mobile Edition. Після подвійного натиснення лівої кнопки миші на цьому файлі, ви побачите вікно з вітанням, натискує на кнопку Next, ви попадете у віконце з ліцензійною угодою. Ознайомившись з угодою і натискує на кнопку Next, на екрані монітора з'явиться діалогове вікно SUN ONE Studio 4 update 1, Mobile Edition Setup.

У цьому і наступному вікні необхідно визначитися з директорією, в якій розміщуватиметься встановлюваний інструментарій, після чого З'ЯВИТЬСЯ діалогове вікно, де перераховуються компоненти установки. Шлях, що задає директорію для установки SUN ONE Studio 4 Mobile Edition не повинен містити пропусків. Вікно вибору компонентів містить елементи для установки. Інсталюється не лише J2ME Wireless Toolkit, але і Java 2 Platform Micro Edition, яка також може поставлятися окремо і лежить на компакт-диску в каталозі \SDK (в тому разі якби вважаєте за краще працювати з командним рядком).

Два подальших вікна носять швидше інформаційний характер, просто натискуйте кнопку Next і майстер установки почне свою роботу по інсталяції середовища програмування SUN ONE Studio .4 Mobile Edition на комп'ютер.

Створення проекту в SUN ONE Studio 4 Mobile Edition

Для створення проекту в середовищі SUN ONE Studio 4 Mobile Edition, необхідно вибрати в меню команду Project => Project Manager, у вікні менеджера проектів, що з'явилося, натискуйте кнопку New.

Замість вікна менеджера проектів з'явиться невелике діалогове вікно Create New Project, де в полі Project Name необхідно задати ім'я проекту, наприклад Demo і натискувати на кнопку ОК. Після цих дій на екрані з'явиться вікно конфігурації створеного проекту.

У вікні конфігуратора проектів є два пункти: Java 2 Standard Edition (J2SE)н Mobile Information Device Profile (CLDC\MIDP), виконані ввиде списку вибраних елементів. Вибравши один з двох пунктів можна конфігурувати проект для створення стандартної програми на Java 2 SE, або додаток для мобільних телефонів якщо вибрати пункт Mobile Information Device Profile (CLDC\MIDP). Вибираємо пункт для створення мобільних застосунків

і натискуємо кнопку Finish. Після чого в нижній частині вікна Explorer, до вкладок Файлові системи, Виконання і Java-doc, додається ще одна вкладка Project Demo. Потім необхідно виробити монтування файлової системи для конфігурованого проекту. У нижній частині вікна Explorer, перейдіть на вкладку ; системи верб ок-, що з'явився

не натискуйте правою кнопкою миші на написі Файлові системи. У відповідь на ці дії з'явиться черга випадних меню, виберіть команду Mount =^ Archive (JAR, Zip). Вибравши команду Archive (JAR, Zip), перейдете до вікна New Wizard -Archive (JAR, Zip), за допомогою якого виробляється монтаж необхідної для проекту бібліотеки Java API. Всі бібліотеки скомпоновані у вигляді архівів, наприклад бібліотека для профілю MIDP 1.0, міститься в архіві midpapi.zip. Ситуація тут декілька заплутана. Як вже говорилося до складу інструментарію SUN ONE Studio 4 Mobile Edition входить також середовище J2ME Wireless Toolkit версії 1.0.4 01, що містить архів API для профілю MIDP 1.0.

При установці пізнішої версії цього інтегрованого середовища ви зможете підключати API і для профілю MIDP 2.0, вибравши відповідний файл з директорії, в яку було встановлене середовище J2ME Wireless Toolkit 2.1 пізнішій версії. У наступній главі буде розглянута установка і налаштування різних телефонних емуляторів і програмного забезпечення для програмування телефонів всіляких марок.

Після монтування файлової системи, необхідно також виробити монтаж директорії, де буде міститися-початковим код створеного проекту. Монтаж директорії відбувається за тією ж схемою, що і монтаж файлової системи, але у випадному меню

потрібно вибрати команду Mount Local Directory і у вікні New Wizard, що з'явилося, - Local Directory вказати місце, де зберігатиметься вихідний код програми, поле чого натискуйте кнопку Finish. Створивши робочий каталог для додатка, його необхідно додати в проект. Натискуйте правою кнопкою миші на тільки що доданий директорії у вкладці Файлові системи вікна Explorer і в меню, що з'явилося, виберете команду Tools => Add to Project.

Створення додатків в SUN ONE Studio 4 Mobile Edition

Завершивши створення проекту, можна переходити до написання коди додатка. Для цього перейдіть на закладку створеного проекту Demo, у вікні Explorer і виберете команду меню File New File, або скористайтесь гарячими клавішами Ctrl+N. На екрані з'явиться вікно New Wizard, де відбувається вибір шаблону створюваного застосування.

Як створювана демонстраційна програма виробимо вибір шаблону HelloMIDlet, який формує код простої програми, що має назву *мідлет* і що позиціює в Java 2 ME як додаток, написаний спеціально для мобільного пристрою. Натискує на кнопку Next, ви попадете в наступне вікно New Wizard - HelloMIDlet, де необхідно вказати майбутню директорію знаходження мідлета і задати ім'я пакету, або скористатися стандартним пакетом, що надається за умовчанням. Вказавши пакет і вибравши директорію, натискуйте на кнопку Finish. Після цього автоматично згенерує простий код прикладу HelloMIDlet проекту Demo, який з'явиться в текстовому редакторі Source Editor.

Компіляція і запуск програми в SUN ONE Studio 4 Mobile Edition

Як демонстраційний приклад ми використовуємо проект Demo, створений автоматично SUN ONE Studio 4 Mobile Edition раніше. Знайдіть вихідний код HelloMIDlet проекту Demo в директорії, яку ви задали для розміщення програм в SUN ONE Studio 4 Mobile Edition або візьміть код HelloMIDlet з компакт-диск диска в теці \Code\HelloMIDlet, помістите його в директорію <C:\WTK21\apps\Demo\src>. Потім натискуйте кнопку Build на панелі інструментів для компіляції і компоновки всього проекту. У робоче вікно SUN ONE Studio 4 Mobile Edition, додадуться дві строчки:

Тут я, звичайно, унеможливлую появи помилок при компіляції, але в реальному програмуванні додатків без цього не обійтися, програміст де-небудь та забуде поставити крапку з комою. Після компіляції і компоновки проекту, натискуйте кнопку Run, і на

екрані з'явиться емулятор за умовчанням DefaultColorPlione, SUN ONE Studio 4 Mobile Edition.

На екран емулятора буде виведено ім'я проекту, натискує на клавішу емулятора Select або Launch, ви попадете в робочий цикл програми, і на екрані емулятора з'явиться напис Test string. Натискує на клавішу Exit можна вийти з додатка.

Упаковка програм

При створенні проекту і подальшої компіляції вихідної коди, у вас з'являться два файли: JAD-файл і файл маніфесту. Якщо ви працюєте з SUN ONE Studio 4 Mobile Edition.; то обоє файлу знаходиться в робочому каталозі проекту.

Файл маніфесту

Файл маніфесту MANIFEST.MF описує можливі атрибути створюваного застосування. При створенні проекту з J2ME Wireless Toolkit в попередніх пунктах згадувалося про діалогове вікно Settings for project «Demo», розділене на сім вкладок, в кожній з яких вказуються різні атрибути створюваного застосування. На основі заданих атрибутів в діалоговому вікні Settings for project «Demo», і відбувається генерація файлу маніфесту, а також JAD-файла.

Файл JAD

Як уже згадувалося, в робочому каталозі проекту «Demo» знаходитиметься 'ще один файл Demo jad. JAD-файл в мобільних застосуваннях ще називають дескриптором додатка (J«v. Application Descriptor). Цей файл використовується телефоном під час роботи програми для здобуття інформації про наявні класи, зображення, піктограми і звукові файли всієї програми. На основі отриманої інформації відбувається управління внутрішніми ресурсами додатка. Якщо ви переміститеся в робочий каталог проекту «Demo» і знайдете файл Demo, що згенерував, jad, то побачите ікону у вигляді телефону з лівого боку від назви файлу. Зробивши подвійне клацання лівою кнопкою миші на файлі Demo.jad, ви запустите емулятор телефону незалежно від того, чи відкрита в даний момент одне з середовищ програмування чи ні. Це ще раз вказує на те, що JAD-файл використовується для управління роботою програми. Але є і ще одна цікава особливість JAD-файлу. JAD-файл теж містить опис атрибутів додатка і вони багато в чому дублюються у файлі маніфесту MANIFEST. MF. У програмах Java 2 ME, JAD-файл - це дескриптор додатка і використовується для управління роботою програми. Сервіс телі

фону перед запуском програми звертається саме до JAD-файлу, визначаючи тим самим можливість роботи з її програми на цьому телефоні. Якщо один з параметрів буде не допустимим для цієї моделі телефону, то додаток не буде запущений.

Файл JAR

У мові Java існує можливість архівації файлів додатка в один файл з розширенням *.jar. Файл JAR - це архів, що містить супутні класи і графічні зображення всього застосування. При написанні програм на Java під різні комп'ютерні операційні системи, програміст вільний сам вибирати, чи буде він поширювати своє застосування у вигляді, що зархівував, або в оригінальному. Ситуація з поширенням-перенесенням програм під мобільні телефони радикально протилежна. Телефони обмежені в своїх ресурсах і в середньому пам'ять, відведена під Java програму, вагається від 30 до 80 кілобайт. Ці цифри обов'язково треба враховувати при створенні мобільних застосувань. Розмір в 40-50 кілобайт вважається оптимальним. У зв'язку з цим всі мобільні програми зобов'язані поширюватися у зархівованому вигляді, тобто в JAR-файлі.

Телефонні емулятори

Інтегровані засоби розробки додатків для мобільних телефонів розглянуті в попередній главі мають в своєму складі декілька емуляторів мобільних пристроїв неіснуючих марок телефонів. Емулятор телефону - це застосування, що програмно емулює роботу реального телефону, враховуючи всі технічні характеристики цього пристрою. Більшість виробників мобільних телефонів мають в своєму арсеналі набори інструментальних засобів для програмування телефонів званих SDK (Software Developer Kit) і безліч вбудовуваних модулів для емуляції певних моделей. Найбільша кількість програмного забезпечення надається компанією Nokia.

Лінійка всіляких моделей телефонів, що випускаються одним виробником, зазвичай ділиться по цінових категоріях. Чим дорожче модель - тим більше функціональних можливостей включено в це пристрій, а значить, передбачена потужніша архітектура телефону. Як правило, всі відмінності зводяться до декількох параметрів:

- ❖ наявність або відсутність операційної системи в телефоні;
- ❖ можливість роботи з додатками написаними на Java 2 ME і C++;
- ❖ кількість пам'яті;
- ❖ роздільна здатність дисплея телефону.

У зв'язку з цим дуже корисно мати якомога більше програмних засобів, що емулюють реальні мобільні пристрої, для того, щоб можна було протестувати створювану програму. Якщо програмувати додаток на Java 2 ME, використовуючи профіль MIDP 2.0 або MIDP 1.0, то на всіх наявних моделях телефонів, де передбачена підтримка Java з відповідним профілем, ваше застосування працюватиме. Але, на жаль, можуть виникнути

проблеми в графічному відображенні елементів призначеного для користувача інтерфейсу, унаслідок того, що присутній ряд очевидних відмінностей у візуальному представленні цих елементів. Також кожен з виробників телефонів дотримується своєї системи навігації, що, безумовно, ускладнює розробку програмного забезпечення. Тому наявні набори SDK і емуляторів окремо взятих моделей телефонів, набагато полегшують роботу програміста.

У своїй більшості програмне забезпечення різних виробників інтегрується в візуальні середовища від компанії Sun Microsystems і Borland, або існують у вигляді окремих програм. Далі буде детальніше вивчення специфіки створення додатків за допомогою класів Java 2 ME.

Зміст звіту

- титульний лист, тема та мета роботи;
- скриншоти з установленими середовищами розробки Java 2 SDK SE, SUN ONE Studio 4 Mobile Edition;
- скриншот з вікном налагодження середовищ та файлу MANIFEST.MF;
- висновки;
- відповіді на контрольні питання.

Контрольні питання

1. Відмінності між Java 2 SDK SE та SUN ONE Studio 4 Mobile Edition.
2. Основні характеристики та призначення середовищ розробки.
3. Як створити проєкт в SUN ONE Studio 4 Mobile Edition?
4. Як виконати компіляцію і запуск програми?
5. Опишіть структуру файлу MANIFEST.MF.
6. Призначення файлів JAD та JAR.
7. Що таке емулятор, навіщо він потрібен.
8. Які характеристики є важливими при обиранні того або іншого пристрою?

ЛАБОРАТОРНА РОБОТА №3.

Механізм роботи додатків Java 2 ME. Мідлети. Створення інтерфейсу користувача.

Мета: озайтися з механізмами роботи мобільних додатків, написати перший мідлет, що реалізує інтерфейс користувача, використовуючи класи, що наводяться в роботі.

Короткі теоретичні відомості

Механізм роботи додатків Java 2 ME

З цього розділу починається безпосередньо опис роботи з кодом. Специфіка Java 2 ME додатків декілька своєрідна, але зовсім не складна. Досить розібратися в загальній моделі побудови програм і все відразу стане зрозуміло.

Мідлет

Додаток, написаний для мобільного телефону, може складатися з різної кількості класів. Одні класи, відповідають за завантаження ресурсів, інші за обробку даних, треті виконують ще якісь додаткові функції, як програміст ви маєте право вибрати будь-яку зручну для вас модель побудови програми. У результаті, створені вами класи, об'єднані в одне ціле, складатимуть одну програму або додаток. Всі застосування, сформовані для роботи в середовищі Java мобільних телефонів, носять назву мідлет. Мідлет - це програма, написана для мобільного телефону з використанням платформи Java 2 ME. Визначати кількість класів програми привілей програміста, але серед всіх класів однієї програми існує один основний клас, з якого починається робота всієї програми. У цьому класі описується код, що відповідає за управління процесом створення інтерфейсу користувача, оголошення набору даних необхідних для роботи всього застосування, створюються об'єкти наявних класів, і що найголовніше, він є відправною крапкою в роботі додатка. Такий клас в Java 2 ME носить назва основний клас мідлета.

Робочі функції, що виконуються цим класом, практично ідентичні методу main (). Пам'ятаєте запис, з яким починався робочий процес додатків написаних на Java 2 SE:

```
public static void main ( String () args )
```

На мобільних пристроях аналогічні дії покладені на підклас класу MIDlet, що виробляє управління робочим процесом всього застосування. Приклад для реалізації поставленого перед вами завдання:

```

/**
Лістинг
Клас HelloMIDlet.java
*/
// імпорт пакетів
import javax.microedition.midlet.*; import javax.microedition.lcdui.* ;
// створюється клас HelloMIDlet
public class HelloMIDlet extends MIDlet implements CommandListener {
// команда виходу з програми private Command exitCommand;
// дисплей телефону private Display mydisplay;
// конструктор класу HelloMIDlet public HelloMIDlet()
{
mydisplay = Display.getDisplay(this);
// вихід з програми
exitCommand = new Command («Вихід», Command.SCREEN, 2);
}
// вхідна точка додатка public void startApp() {
// об'єкт класу TextBox
TextBox t = new TextBox ("HelloMIDlet", "Текст", 256, 0);
// додає команду виходу t.addCommand(exitCommand);
// виставляє обробник подій для команди виходу
t.setCommandListener(this); // відображає поточний екран телефону
mydisplay.setCurrent(t);
}
// пауза в роботі додатка public void pauseApp() { // вихід з програми
public void destroyApp (boolean unconditional) { // обробник подій для
команд додатка public void CommandAction (Command c, Displayable s) {
// обробка команди вихід if. (c == exitCommand) {
// звільняє ресурси і вивантажує з пам'яті міدلлет
destroyApp(false); notifyDestroyed();
}
} }
}

```

Перше, що впадає в очі при детальному розгляді коду міدلлету - це три методи: startApp(), pauseApp() і destroyApp(). Програмісти, які знайомі з аплетами, знайдуть явну схожість в структурі міدلлета і аплета, навіть назви в деякій мірі схожі. У аплете також є подібні методи: start(), stop() і destroy(). Можна сказати, що основний клас міدلлета - це деякий симбіоз аплета і методу main(), що грає основну роль, що управляє, в додатку для мобільних пристроїв. Створюючи свої класи і реалізуючи їх код, ви повинні покласти основні робочі функції на основний клас міدلлета.

Інтерфейс користувача

Коли ми розглядали механізм роботи прикладу з попереднього лістингу, я думаю, ви помітили деякий лозкранный принцип відображення інформації на дисплеї. Перший екран показував список доступних застосувань, після вибору одного з них ви потрапляли на екран цього застосування. Натискує кнопку Вихід, відбувалося повернення до екрану вибору. У Java 2 ME програмах така схема позкранного відображення інформації є основною. Якщо ви ніколи не звертає уваги на це при роботі зі своїм телефоном, то самий час узяти його і поклацувати джойстиком. У додатках для мобільних телефонів заснованих на екранах в Java 2 ME відсутні вікна і фрейми на відміну від Java 2 SE. Телефони обмежені в системних ресурсах і всіляка красива розкіш, на жаль не дозволена для цих маленьких пристроїв. Тому при створенні кінцевого продукту варто з особливою ретельністю продумувати основні складові майбутнього застосування. Грунтуючись на позкранном відображенні інформації, необхідно створювати інтуїтивно зрозумілу структуру додатка, утворюючи при цьому чітку екранну навігацію. Якщо користувач заблукає у вашій програмі, він просто видалить її з пам'яті телефону і ніколи до неї більше не повернеться, а ви втратите потенційного покупця.

Як вже наголошувалося, екран телефону представлений класом `Display`. Кожен мидлет може мати лише один об'єкт класу `Display`, повертаний мидлету за допомогою методу `getDisplay()`, визначаючи тим самим поточний дисплей телефону для мидлета.

Платформа Java 2 ME володіє пакетом `javax.microedition.lcdui`, що включає класи для роботи з призначеним для користувача інтерфейсом UI (user interface). Велика кількість класів, що входять в цей пакет, детально розглядатимуться в наступній главі. Найголовнішим класом призначеного для користувача інтерфейсу є клас `Displayable`. З основи абстрактного класу `Displayable` відбувається побудова основної частини графічного інтерфейсу додатка.

Від класу диспетчера `Display` залежить, який з класів `Displayable` відображатиме на екрані. У свою чергу лише один клас `Displayable` може бути одноразово показаний на екрані. Тобто, об'єкт класу `TextBox`, грубо кажучи, існує в своєму екрані, об'єкт класу `List` - в своєму і обоє об'єкту не можуть існувати разом на

одному екрані, визначаючи тим самим правило позкранного віддзеркалення інформації на дисплеї телефону.

Далі в ієрархії структури призначеного для користувача інтерфейсу, йдуть два абстрактні класи: Screen-и Canvas. На цій стадії відбувається розділення класів призначеного для користувача інтерфейсу на високорівневий клас, призначений класу Screen і всій його подальшій ієрархії спадкоємства і низькорівневий клас Canvas. Обидва класи створюють структуру інтерфейсів, розділену на високорівневий і низькорівневий призначений для користувача інтерфейси.

Високорівневий інтерфейс містить засоби для роботи з призначеним для користувача інтерфейсом, створеним на основі класів шаблонів, використання яких наводить до побудови жорстко заданого інтерфейсу. Наприклад, задіяний у вихідному коді HelloMIDlet проекту Demo клас TextBox, не може жодним чином змінити екран телефону. Екран представлений класом TextBox - це текстовий контейнер, в якому можна здійснювати вивід, видалення і редакцію тексту і не більш того. Тобто класи високорівневого інтерфейсу - це жорстко задана модель відображення призначеного для користувача інтерфейсу на екрані телефону, за допомогою яких програміст організовує навігацію, списки, меню, текстові контейнери, групи вибраних елементів і так далі.

Низькорівневий інтерфейс надає графічні засоби для малювання на екрані різних графічних елементів і обробку команд, що посилаються з клавіш телефону. Низькорівневий інтерфейс дозволяє малювати на екрані телефону, додаючи тим самим в додаток красиві графічні елементи у вигляді таблиці, зображень, полів, заставок і такдале. Таке розділення класів існує суто в теоретичному вигляді і ніщо вам не заважає комбінувати елементи обох інтерфейсів в одній програмі, створюючи красиве інтерактивне застосування.

Розглянемо деякі характеристики класів Alert, TextBox, Formn List, що представляють високорівневий інтерфейс:

Alert - надає можливість в створенні повідомлень або повідомлень про помилки у вигляді окремих екранів; Q TextBox - масив даних містить текстову інформацію з можливістю її редакції; Q Form - екранний контейнер, в якому можна розмістити різні елементи інтерфейсу за допомогою додаткових підкласів класу Item; Q List - список елементів, що дозволяє виробляти вибір різних операцій.

Далі вивчатимуться класи високорівневого інтерфейсу, за допомогою яких створюються списки, групи елементів, текстові поля і безліч інших елементів призначеного для користувача інтерфейсу.

Класи призначені для створення інтерфейсу користувача

У Java 2 ME є пакет `javax.microedition.lcdui`, визначений для класів призначеного для користувача інтерфейсу. Як вже наголошувалося в главі 5, класи призначеного для користувача інтерфейсу розділені на високорівневий і низькорівневий інтерфейси. У цій главі будуть послідовно розглянуті всі класи високорівневого призначеного для користувача інтерфейсу. Кожен з розділів містить інформацію про один конкретний клас, що надає ряд можливостей в оформленні інтерфейсу користувача. Використовуючи можливості цих класів, ви зможете створювати в додатку списки, групи елементів, завантажувати в програму зображення, використовувати рядок, що біжить, призначати шрифт тексту і багато що інше. У розділах по кожному класу призначеного для користувача інтерфейсу, аналізуються конструктори, основні методи і константи класу, за допомогою яких в кінці кожного розділу створюється додаток, що ілюструє роботу цього класу", і наводиться лістинг всієї програми. При розгляді методів і констант класів використовується лише основні компоненти. Для детального аналізу складових одного з класів звернетеся до додатка 2, яке виконане у вигляді довідника по Java 2 ME.

Єдине, про що необхідно пам'ятати при проектуванні і створенні призначеного для користувача інтерфейсу додатка - це про використовуваний профіль MIDP. У деякі класи, що є у складі профілю MIDP 1.0, були додані нові методи, константи і навіть класи з профілю MIDP 2.0. Наприклад, в класі `ChoiceGroup` для профілю MIDP 1.0 доступні тринадцять методів, а вже в профілі MIDP 2.0 їх налічується сімнадцять, тобто, додано ще чотири нових. У тій же документації по Java 2 ME якоїсь чіткої грані що розділяє два профілі не існує, але при розгляді методів і констант згадується про приналежність до одного з профілів. Тому при створенні додатків, наприклад під профіль, MIDP 1.0, необхідно уважно планувати розробку програм і не задіювати компоненти профілю MIDP 2.0. Якщо ви розробляєте програму для профілю MIDP 2.0, то можете користуватися всіма наявними компонентами незалежно від приналежності до одного з профілів.

Клас Form

Основним екранним класом прикладів з глави 5 служив екран, представлений класом `Form`. Як ви розумієте цю не обов'язкову умову, але я вибрав клас `Form` не

випадково. Річ у тому, що реалізація класу `Form` виконана у вигляді контейнера, що дозволяє вбудовувати в себе різні компоненти призначеного для користувача інтерфейсу. Це, мабуть, єдиний і найпотужніший по своїх можливостях клас. Само по собі назва `Form` має на увазі створення деякої форми для заповнення екрану телефону. У попередніх прикладах ми використовували, так звану, порожню форму цього класу у вигляді чистого екрану. Згодом, при згадці терміну форма, матиметься на увазі екран телефону, представлений об'єктом класу `Form` для інтеграції класів призначеного для користувача інтерфейсу.

Клас `Form` має два конструктори, необхідних при створенні об'єкту цього класу. Першим конструктором ми вже користувалися, і виглядає він досить просто:

```
public Form (String title)
```

Параметри конструктора класу `Form`:

- ❖ `title` - заголовок з'являється у верхній частині створеного вікна. Другий конструктор класу `Form` має вже два параметри, і дозволяє

вбудувати компоненти інтерфейсу в порожню форму, `public Form (String title, Item[] items)`

Параметри конструктора класу `Form`:

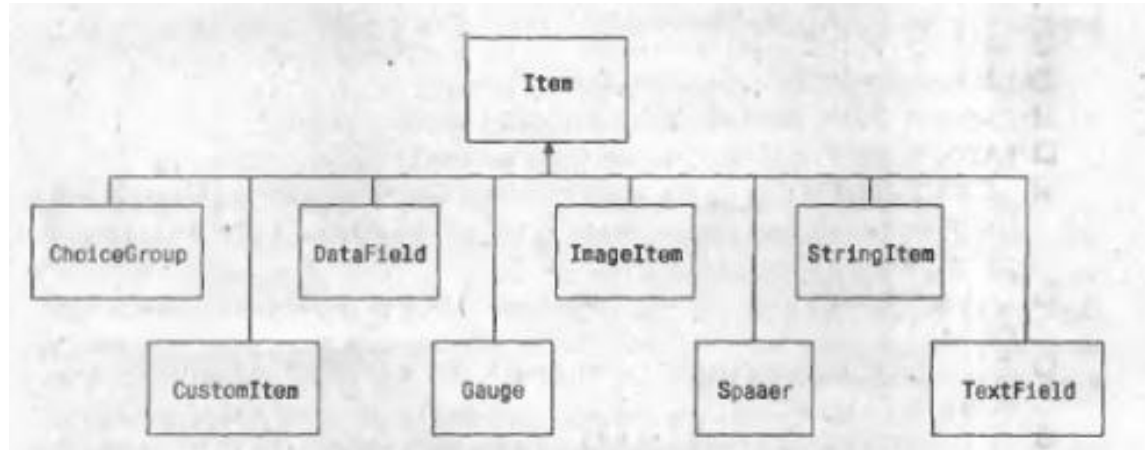
- ❖ `title` — заголовок вікна;
- ❖ `items` - масив компонентів розміщуваних в класі `Form`.

У класі `Form` існує набір методів, за допомогою яких можна додати, видалити або вставити компоненти інтерфейсу. Клас `Form` має дванадцять методів, за допомогою яких можна маніпулювати компонентами абстрактного класу `Item`. Всього налічується вісім компонентів призначеного для користувача інтерфейсу в ієрархії класу `Item`. Тобто ви створюєте екран, за який відповідає клас `Form`, і інтегруєте наявні у вашому розпорядженні компоненти класу `Item`. Про сам клас `Item` і його ієрархії ми поговоримо в наступному розділі цієї глави, після аналізу класу `Form`.

Клас `Item`

Абстрактний суперклас `Item` має ієрархію з восьми підкласів. Кожен підклас представляє один з елементів призначеного для користувача інтерфейсу, наприклад, клас

TextField, створює текстові поля ДЛЯ введення пароля, адреси електронної пошти або просто числових значень. Всі вісім класів, по суті, встановлюють компоненти призначеного для користувача інтерфейсу, які вбудовуються у форму визначену класом Form. На рис змальована ієрархія абстрактного суперкласу Item.



Ієрархія суперкласу Item

Choice Group - це група зв'язаних елементів для подальшого вибо-. ра передбачуваних дій;

Про CustomItem - за допомогою цього класу можна додавати різні графічні елементи у форму;

ID Date Field - клас, за допомогою якого є можливість редагувати час і дату;

Про Gauge - допускає графічне відображення діаграм, процесів завантаження;

Про ImageItem - здійснює показ зображення на екрані телефону;

Spacer - задає визначене по розміру простір;

String Item за допомогою цього класу можна створити довільний текст. Цей клас не допускає редагування, він лише відображує інформацію;

TextField - надає текстові поля для редакції. Любий із розглянутих класів успадковується з суперкласу Item і може бути доданий на екран, створений класом Form. Кожен компонент класу Item містить з лівого боку область, де за бажання можна відображувати зображення у вигляді ікони. При переміщенні компонента, ікона також переміщається разом з компонентом. Клас Item за допомогою наявних в його складі директив задає, в основному, формат відображення для будь-якого компонента. Формат

визначає задану ширину, висоту або вирівнювання компонентів у формі, а також клас `Item` має безліч методів що здійснюють контроль над компонентами.

Клас `Alert`

Використання класу `Alert` в Java 2 ME додатках обумовлено виникненням різних позаштатних ситуацій. В основному клас `Alert` застосовується для створення екрану, який інформує користувача про помилку такою, що сталася в додатку або будь-якому іншому повідомленні інформаційного, характеру. Екран, визначений класом `Alert` може містити строкове повідомлення про помилку, що сталася, або текстовий рядок із заданим зображенням. У зв'язку з цим, клас `Alert` має два, конструктора, об'єктів цього класу, що використовуються в створенні. Перший конструктор містить один параметр типу `String`, задаючи рядок тексту для повідомлення. Розглянемо перший конструктор класу `Alert`.

```
public Alert (String title);
```

Параметри конструктора `public Alert` :

- v title - рядок тексту.

Другий конструктор класу `Alert` має вже чотири параметри, представляючи цікавіший вигляд створюваного екрану.

```
public Alert (String title, String alertText, Image alertImage, AlertType alertType)
```

Параметри конструктора `public Alert` :

- v title - назва створеного екрану;

- v alertText - текст повідомлення;

- v alertImage - зображення;

- v alertType - тип повідомлення, визначуваний класом `AlertType`. Існує п'ять типів повідомлень:

- v static `AlertType ALARM` - тривога;

Про static `AlertType CONFIRMATION` - попередження про можливу дію, яку користувач повинен виробити;

- v static `AlertType ERROR` - помилка;

- v static AlertType INFO - інформаційне повідомлення ;
- v static AlertType WARNING - попередження.

Створюючи об'єкт класу Alert, ви можете вибрати необхідного типа повідомлень або інформаційних повідомлень, формуючи органічні, зручні застосування, що передбачають будь-які варіанти розвитку подій.

Зміст звіту

- титульний лист, тема та мета роботи;
- узгодження з викладачем тема для створення інтерфейсу користувача;
- текст програми з функціями класів;
- скриншот з вікном інтерфейсу користувача;
- висновки;
- відповіді на контрольні питання.

Контрольні питання

9. Відмінності між Java 2 SDK SE та SUN ONE Studio 4 Mobile Edition.
10. Дайте визначення інтерфейсу користувача.
11. Що такк мідлети? Приклади.
12. Які основні класи ви знаєте для створення інтерфейсу користувача?
13. Класи: Displayable, Alert, TextBox, Formn List, Form, Item.
14. Відмінності та призначення вищезначених класів.

ЛАБОРАТОРНА РОБОТА №4.

Використання графіки в мобільних телефонах. Технологія створення ігор.

Мета: навчитися використовувати графіку в мобільних додатках, написати програму гри з використанням графіки.

Короткі теоретичні відомості

Програмування графіки та техніка створення ігор

Високорівневі класи, вивчені в попередній главі, дають можливість створювати призначений для користувача інтерфейс додатка. По суті, ці класи виконані у вигляді шаблонів, використовуючи які ви можете створювати списки, форми, шрифт, групи елементів, рядки, що біжать. Але використання таких класів-шаблонів декілька спрощує інтерфейс програми, позбавляючи можливості використання графіки в програмах на Java2ME. Інколи в додатку необхідно намалювати таблицю, лінію, квадрат, тобто скористатися графікою для створення насиченої і барвистої програми. Для етих цілей в Платформі Java2ME існують так звані класи низькорівневого інтерфейсу - це класи Canvas і GameCanvas (клас GameCanvas буде розглянутий в наступній главі), а так само клас Graphics, за допомогою якого здійснюється безпосереднє промальовування графіки на екрані телефону.

Ринок мобільних телефонів розвивається стрімкими темпами. Все більше телефонів мають підтримку технології Java. Віяння ігрової індустрії захопило і мобільні телефони, тому платформа Java 2 ME позиціює більшою мірою як ігрова платформа для мобільних телефонів. При розробці ігор під профіль MIDP 1.0 програміст стикається з масою проблем у вигляді написання великої кількості власних класів для створення ігрового процесу, малювання графіки, рівнів і так далі. У профіль MIDP 2.0 додано п'ять ігрових класів, мобільних ігор, що значно спрощують створення, це:

- ❖ Game Canvas - абстрактний клас, що містить основні елементи ігрового інтерфейсу;
- ❖ Layer - абстрактний клас, що відповідає за рівні що представляються в грі;
- ❖ LayerManager - менеджер рівнів;
- ❖ Sprite - представляє на екрані спрайтове зображення;
- ❖ TiledLayer- відповідає за створення фонових зображень.

Всі вищеперелічені класи доступні лише в профілі MIDP 2.0. Також можна скористатися цими класами і в програмуванні телефонів Siemens під профіль MIDP 1.0, імпортувавши пакет `com.siemens.mp.color_game`. Компанія Siemens входить до експертної групи MIDP Expert Group і, видно, на базі вже наявних ігрових класів від компанії Siemens були створені ігрові класи для профілю MIDP 2.0.

При використанні ігрових класів профілю MIDP 2.0, побудова мобільної гри заснована на системі рівнів. Формуючи гру, програміст створює необхідне йому кількість рівнів. Кожен з рівнів містить набір графічних елементів, наприклад, можна створити рівень з фоновим зображенням, рівень з перешкодами, рівень з ігровими бонусами, рівень з головним персонажем гри. Після цього все наявні рівні компонуються воедино, накладаються один рівень на інший і промальовувалися на екрані телефону. Контроль над всіма рівнями, здійснюється за допомогою класу `LayerManager` -менеджера рівнів. Велика кількість методів що надаються ігровими класами дозволяє відстежувати всілякі зіткнення, переміщення, анімацію і безліч інших функцій, що значно спрощує процес створення мобільних ігор.

У цій главі, спочатку ви познайомитеся з пристроєм всіх ігрових класів, а потім буде створений ряд прикладів, що ілюструє роботу і взаємодію ігрових класів профілю MIDP 2.0.

Клас Layer

Абстрактний клас `Layer` задає основні властивості для всіх створених рівнів гри.

Клас `Layer` має два підкласу `TiledLayer` і `Sprite`. При створенні будь-яких інших підкласів класу `Layer` необхідно реалізувати метод `paint ()` у цих класах, аби мати можливість малювати створені рівні на екрані телефону, що представляється класом `Graphics`. За допомогою методів класу `Layer` можна задавати розмір і позицію рівня.

- ❖ `int getHeight ()` - отримує висоту екрану;
- ❖ `int getWidth()` - отримує ширину екрану;
- ❖ `int getX()` - отримує горизонтальну координату рівня;
- ❖ `int getY ()` - отримує вертикальну координату рівня;
- ❖ `void move (int dx, int dy)` – переміщує рівень на `dx` и `dy` к; `Q`
- `abstract void paint (Graphics g)` - малює рівень;
- ❖ `void setPosition (int x, int y)` - встановлює рівень в позицію, позначену в координатах `x` і `y`.

Для створення об'єкту потрібно скористатися конструктором класу `LayerManager`.

`G LayerManager ()` - створює рівень.

А щоб додати рівні в гру необхідно використовувати наступні методи:

`O void append (Layer 1)` - додає рівень в менеджер рівнів;

`Q Layer getLayerAt (int index)` - отримує рівень із заданим індексом;

Клас TiledLayer

За допомогою класу `TiledLayer` створюється фон ігрової сцени.

`int getSize ()` - отримує загальну кількість рівнів;

`void insert (Layer 1, int index)` - підключає новий рівень в заданий індекс;

`void paint (Graphics g, int x, int y)` - представляє поточний менеджер рівнів в заданих координатах;

`void remove (Layer 1)` - видаляє рівень з менеджера рівнів.

Передбачимо, у вас є чотири рівні: фон, гравець, перешкоди і артефакти. Для того, щоб зв'язати всі чотири рівні воедино, створюється об'єкт класу `LayerManager` і визивається метод `append ()`. Наприклад:

```
La'yerManager lm = new LayerManager ();
```

```
lm.append(fon);
```

```
lm.append(igrok);
```

```
lm.append(prepare);
```

```
lm.append(artf);
```

Тепер чотири перераховані рівні відбивається на ігровому полі.

Клас Sprite

Механізм роботи з об'єктом класу `Sprite` ідентичний моделі роботи з класом `TiledLayer`. Але якщо клас `TiledLayer` в основному відповідає за фонове зображення, то за допомогою класу `Sprite` малюються на екрані основні анімовані герої, космічні кораблі, машини, люди, звірі, артефакти і так далі. Зображення, що завантажується в гру, може бути свиконане у вигляді анімаційної послідовності. Кількість малюнків в анімаційній

послідовності необмежено, а відлік походить від нуля. Розташовуватися малюнки можуть як у вигляді стовпця, так і у вигляді колонки, залежно від ваших переваг. Кожен малюнок анімаційної послідовності називається фреймом. Фрейм може бути будь-якого розміру по ширині і висоті, але всі фрейми мають бути однакових розмірів. Розмір фрейму має бути відомий, тому що він використовується при створенні об'єкту класу Sprite. Є три конструктори класу Sprite кожен з яких можна використовувати при створенні об'єкту класу Sprite.

- ❖ `Sprite (Image image)` - створює не анімований спрайт;
- ❖ `Sprite(Image image, int frameWidth, int frameHeight)` - створює анімаційний спрайт, узятий із заданого фрейма;
- ❖ `O Sprite(Sprite s)` - створює спрайт з іншого спрайту.

Для переходу по фреймах вихідного зображення, а також визначення зіткнення між об'єктами використовуються методи класу Sprite.

- ❖ `boolean collidesWith(Sprite s, boolean pixelLevel)` - визначає зіткнення між спрайтом;
- ❖ `boolean collidesWith(TiledLayer t, boolean pixelLevel)` -определяет зіткнення між спрайтом і перешкодою, намальованою за допомогою `KnaccaTiledLayer`;
- ❖ `int getFrame ()` - отримує поточний фрейм;
- ❖ `void nextFrame ()` - здійснює перехід на наступний фрейм;
- ❖ `void paint (Graphics g)` - малює спрайт;
- ❖ `void prevFrame ()` - здійснює перехід, на попередній фрейм;

`O void setFrame(int sequenceIndex)` - встановлює заданий фрейм;

- ❖ `void setFrameSequence (int [ ] sequence)` - встановлює певну фреймову послідовність;
- ❖ `O void setImage(Image img, int frameWidth, int frameHeight)` - змінює зображення спрайту на нове зображення;
- ❖ `O void setTransform (int transform)` - виробляє трансформацію спрайту.

`Q public void defineReferencePixel(int x, int y)` – змінює опорну позицію спрайту, переносячи її в крапку з координатами x і y .

Метод `defineReferencePixel()` змінює опорну позицію спрайту, але для чого це потрібно? Опорна позиція для спрайту задається лівим верхнім кутом, але не завжди це

зручно, тому опорну позицію можна перенести, в центр спрайту. Наприклад, якщо спрайт зроблений у вигляді машини, то при обертанні довкола своєї осі, якщо опорна позиція перенесена в центр, обертання буде правдоподібним. Але якщо не перевизначити позицію, то обертання станеться в точці лівого верхнього кута спрайту і виглядати це буде не цілком природно, неначе в машини пробите одне з коліс. Для цього викликається метод `defineReferencePixel ()` із заданими координатами і перевизначає опорну позицію, наприклад в центр спрайту:

```
defineReferencePixel(frameWidth / 2, frameHeight / 2);
```

Метод `setTransform ()` виробляє трансформацію спрайту, обертаючи або дзеркально відображуючи спрайт довкола своєї осі за допомогою наступних констант:

- ❖ `static int TRANS_MIRROR;`
- ❖ `static int TRANS_MIRROR_ROT180;`
- ❖ `static int TRANS_MIRROR_ROT270;`
- ❖ `static int TRANS_MIRROR_ROT90;`
- ❖ `static int TRANS_NONE;`
- ❖ `static int TRANS_ROT180;`
- ❖ `static int TRANS_ROT270;`
- ❖ `static int TRANS_ROT90.`

Розглянувши класи `GameCanvas`, `Layer`, `Sprite`, `TiledLayer` і `Layer-Manager`, зробимо їх невеличке застосування.

Створення фонового зображення

За допомогою класу `TiledLayer` можна створювати будь-яку кількість рівнів, накладаючи їх один на одного, а за допомогою менеджера рівнів, представленого

класом `LayerManager`, відстежувати всі наявні рівні. Як приклад буде створений фон на основі елементів розмітки ігрового поля. Фонове зображення завантажується з файлу `fon.png`. Само зображення виконане у вигляді шести вічок розміром 15x15 пікселями.

У лістингу знаходиться код прикладу того, що створює фонове зображення. Ознайомтеся з лістингом, а потім ми перейдемо до аналізу цього прикладу.

```
/** лістинг клас MainGame */
```

```

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class MainGame extends MIDlet implements CommandListener {
// команда виходу
private Command exitMidlet = new Command('Вихід',Command.EXIT, 0);
// об'єкт класу MyGameCanvas
private MyGameCanvas mr;
public void startApp() {
// обробляємо виняткову ситуацію try{
// ініціалізували об'єкт класу MyGameCanvas mr = new MyGameCanvas();
// запускаємо потік mr.start ();
// додаємо команду виходу mr.addCommand(exitMidlet); mr.setCommandListener(this);
// відображаємо поточний дисплей
Display.getDisplay(this).setCurrent(mr);
}catch (java.io.IOException ex) {}
}
public void pauseApp () {}
public void destroyApp (boolean unconditional )
{
// зупиняємо потік if(mr != null) mr.stop();
}
public void commandAction(Command c, Displayable d){
if (c == exitMidlet) {
destroyApp(false); notifyDestroyed();
}
}
}
/**
Файл MyGameCanvas.java
клас MyGameCanvas */
import java.io.IOException; import javax.microedition.lcdui.*; import
javax.microedition.lcdui.game.*;
public class MyGameCanvas extends GameCanvas implements Runnable {
// створюємо об'єкт класу TiledLayer private TiledLayer fonPole; // створюємо об'єкт класу
LayerManager private LayerManager lm;
// логічна змінна для виходу з циклу boolean z;
public MyGameCanvas() throws IOException {
// звертаємося до конструктора суперкласу Canvas super(true);
// ініціалізували fonPole fonPole = Fon(); // створюємо менеджер рівнів
lm = new LayerManager ();

```

Зміст звіту

- титульний лист, тема та мета роботи;
- узгодження з викладачем тема для створення гри;

- текст програми з функціями класів;
- скриншот з вікном гри;
- висновки;
- відповіді на контрольні питання.

Контрольні питання

1. Що таке графіка? Особливості графіки у мобільних пристроях.
2. Які класи слід використовувати для виводу графіки?
3. Відмінності класів Layer, TiledLayer, Sprite.
4. Технологія створення гри.
5. Що таке фонове зображення? Як його використовувати у мобільних пристроях?.
6. Що виконує програма, яка є в прикладі лістингу?

Висновки

У роботах були з'ясовані основні конфігурації і профілі платформи Java 2 Micro Edition, було розглянуто дві версії профілів MIDP 1.0 і MIDP 2.0, вивчено основи побудови мобільних програм, використання графіки, техніка створення ігор.

Отже, Java - новітня об'єктно-орієнтована мова, легка у вивченні і дає можливість створювати програми, які можуть виконуватися на будь-якій платформі без яких-небудь доопрацювань (кроссплатформенність).

Платформа Java 2 ME складається з віртуальної машини Java, мови програмування Java і величезної кількості бібліотек, що полегшують роботу програміста. Мова програмування Java працює на основі інтерпретації, тому для роботи програм написаних на цій мові необхідна віртуальна машина Java, вбудована в пристрій, на якому планується запускати Java програми. Наявність віртуальної машини Java в мобільному телефоні означає, що виробник підтримує технологію Java. В даний момент практично всі виробники мають підтримку Java технології, що дозволяє писати будь-які програми цією мовою і завантажувати їх в телефон.

Java 2 ME - це безумовно стандарт, що забезпечує сумісність програмного забезпечення з будь-якою маркою телефону. Стрімкий розвиток технологій мобільних пристроїв зобов'язаний мати на увазі певний еталон у даному напрямі. Чимала кількість сторонніх розробників при строгій стандартизації зацікавлена в розвитку ринку програмного забезпечення телефонів. У наш час телефон має майже кожна людина, у відмінність, скажемо від портативних комп'ютерів на основі операційних систем Palm і Pocket PC. Тому ринок мобільних пристроїв по своїх масштабах вважається найбільш перспективним, а платформа Java 2 ME, є стандартом у створенні програм для телефонів.

Список використаних джерел

1. Горнаков С. Г. Программирование мобильных телефонов на Java 2 ME — ДМК прес, Москва, 2005 р, - 311 с.
2. <http://java.sun.com/> – The Source for Java Developers.
3. <http://www.intuit.ru/department/pl/javapl/> – Программирование на Java.
4. <http://javaland.com.ua/> – Програмування на Java
5. <http://uk.wikipedia.org/wiki/Java/> – Мова програмування Java.