Язык программирования JAVA

Введение
Базовый синтаксис и типы данных
Управление выполнением программы

Содержание курса

- Введение. Базовый синтаксис и типы данных. Управление выполнением программ.
- Создание собственных классов. Работа со строками. Внутренние классы.
- Наследование и полиморфизм. Массивы и коллекции.
- Абстрактные классы и интерфейсы.
- Обработка исключений.
- Создание интерфейса пользователя.
- Работа с потоками ввода/вывода.
- Мультизадачность в Java.

Что такое JAVA?

- Объектно-ориентированный язык программирования
- Изначально был разработан для управления бытовой электроникой
- Поставляется с большой библиотекой классов
- Использует виртуальную машину (JVM) для выполнения программ

Ключевые особенности JAVA

- Объектно-ориентированный
- Интерпретируемый и платформонезависимый
- Динамическая загрузка библиотек
- Мультипоточность
- Надежность и безопасность

Объектно-ориентированный подход

- Объекты и классы
  - Объект-представление «вещи» в реальном мире
  - Класс – «шаблон», определяющий «вещи»
- Модель классов объединяет
  - Существующие классы и объекты
  - Поведение, цели и структуру
  - Отношения между классами
  - Отношения между объектами
- Модель используется во всем проекте

Независимость от платформы

- Исходные тексты хранятся в текстовом виде в файле .java
- Файл .java компилируется в файл .class
- Этот файл содержит байт-код (инструкции для выполнения интерпретатором)
- Байт-код интерпретируется во время выполнения

Just-In-Time (JIT) компилятор

- Компилирует байт-код в исполняемый код для конкретной платформы
- Увеличивает производительность
- Оптимизирует повторяющийся код, например, циклы

Java - приложения

- Клиентские
  - JVM выполняет отдельное приложение из командной строки
  - Классы загружаются с локального диска
- Серверные
  - Обслуживают несколько клиентов
  - Применяются для многозвенных приложений

Java - апплеты

- Предназначены для встраивания в HTML страницы
- Выполняются внутри браузера и могут взаимодействовать с сервером
- Могут быть преобразованы в обычное приложение

Java SDK (JDK)

- Sun Java SDK включает в себя
  - Компилятор (javac)
  - Библиотеку классов
  - Отладчик (jdb)
  - Интерпретатор (java)
  - Генератор документации (javadoc)
  - Архиватор (jar)
  - Другое…

Варианты поставки

- J2ME (Micro Edition) – для мобильных устройств
- J2SE (Standard Edition) – разработка обычных приложений
- J2EE (Enterprise Edition) – разработка приложений многозвенной архитектуры

Инструменты, используемые в данном курсе

- J2SE версии 1.4.2
- В качестве IDE будет использоваться Eclipse

Основы объектно-ориентированного программирования

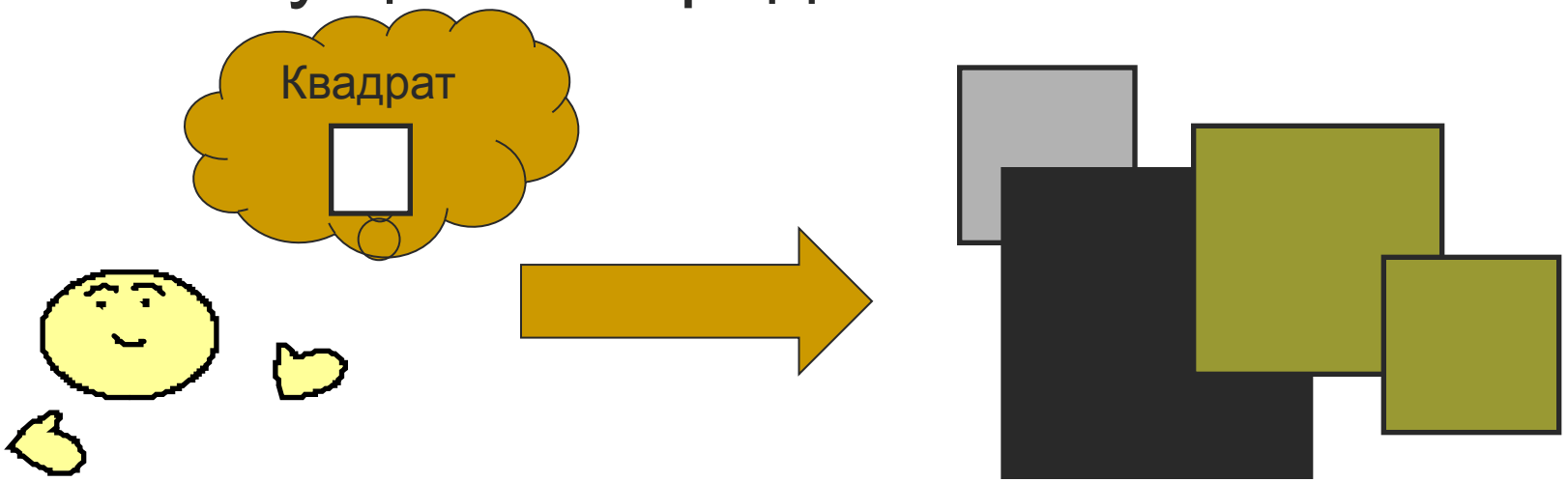
- Объектная модель
- Объекты и классы
- Взаимодействие классов

Зачем нужна объектная модель?

- Единое представление об окружающем мире в одном проекте
- Простота модификации
- Расширяемость

Объекты и классы

- Объект – некоторая КОНКРЕТНАЯ сущность моделируемой предметной области
- Класс – шаблон или АБСТРАКЦИЯ сущности предметной области



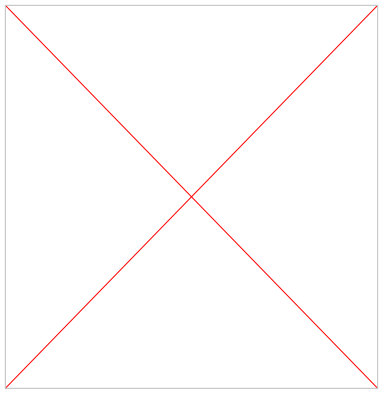
Свойства классов и объектов

- Уникальные характеристики, которые необходимы при моделировании предметной области
- ОБЪЕКТЫ различаются значениями свойств
- Свойства отражают состояние объекта

Методы классов и объектов

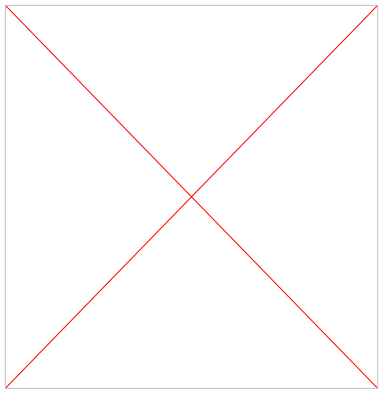
- Метод отражает ПОВЕДЕНИЕ объектов
- Выполнение методов, как правило, меняет значение свойств
- Поведение объекта может меняться в зависимости от состояния

Инкапсуляция



- Значение свойств можно менять ТОЛЬКО ПОСРЕДСТВОМ ВЫЗОВА МЕТОДОВ

Наследование

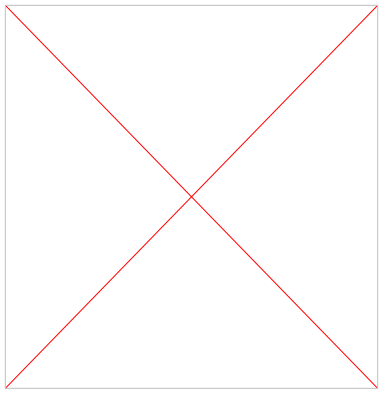


- Класс может иметь одного или нескольких потомков. Потомки (подклассы) содержат в себе тот же набор свойств и методов, что и класс-родитель (суперкласс).

Наследование

- В Java нет множественного наследования
- Все методы в Java – виртуальные (могут быть перекрыты), если явно не указано обратное

Полиморфизм



- Технология, позволяющая задать для одного и того же (по смыслу) метода различные способы выполнения, в зависимости от класса, в котором этот метод реализован
- Основан на наследовании
- Методы ОБЯЗАНЫ иметь одно и то же имя и набор параметров (сигнатуру)

Взаимодействие классов

- Зависимость (uses-a) – класс использует другой класс
- Агрегирование (has-a) – класс содержит объекты другого класса
- Наследование (is-a) – класс является наследником (родителем) другого класса

Итоги

- Java – объектно-ориентированный, платформонезависимый язык программирования
- Выполняется в виртуальной Java машине (JVM)
- Программы, написанные на Java, могут выполняться внутри HTML страниц (апплеты)

Базовый синтаксис языка

- Ключевые компоненты SDK
- Структура файла
- Определение классов
- Базовые типы данных и операторы
- Использование переменной CLASSPATH
- Компиляция и запуск приложения

Ключевые компоненты SDK

- Компилятор (javac) – создает из исходного кода байт-код
- Интерпретатор (java) – выполняет байт-код

Пакеты

- Классы объединяются в специальные структуры, называемые пакетами
- Стандартные пакеты для
  - Поддержки базовых конструкций языка (java.lang)
  - Создания оконного интерфейса (javax.swing)
  - Управления вводом/выводом (java.io)

Структура исходного файла класса Java

- Исходный файл состоит из следующих частей
  - Необязательное слово **`package`**, за которым следует наименование пакета, в котором содержится класс
  - Необязательный оператор **`import`** (может быть несколько), который указывает, какие классы из сторонних пакетов используются создаваемым классом
  - Одно или более определение **`class`** или **`interface`**, за которым следует программный блок
  - Файл должен иметь ТО ЖЕ имя, что и создаваемый класс
- Ключевые слова языка Java ЧУВСТВИТЕЛЬНЫ К РЕГИСТРУ
- В файле может быть ТОЛЬКО ОДИН public класс

Пример класса Java

```
package ru.vsu.test;
import java.util.Date;
public class FirstProgram {
    private Date today;
    public Date getToday(){
        return today;
    }
    public void setToday(Date aToday){
        today = aToday;
    }
    public static void main (String[] args){
        FirstProgram fp = new FirstProgram();
        fp.setToday(new Date());
        System.out.println (fp.getToday());
    }
}
```

Соглашения об именовании

- **Имена файлов**
  - **Customer.java**
  - **Person.class**
- **Имена пакетов**
  - **java.util**
  - **javax.swing**
- **Имена классов**
  - **Customer**
  - **Person**
- **Имена свойств класса**
  - **firstName**
  - **id**
- **Имена методов**
  - **getName**
  - **isAlive**
- **Имена констант**
  - **SQUARE_SIZE**

Также могут использоваться цифры 1..9, _, $

Определение класса

- Определение класса включает:
  - Модификатор доступа
  - Ключевое слово class
  - Свойства класса
  - Конструкторы
  - Методы
  - Статические свойства
  - Статические методы

Пример

```
public class FirstProgram {
    private Date today;
    public Date getToday(){
      return today;
    }
    public static final PROGRAM_SIZE=560;
    public static void main (String[] args){
    ...
    }
}
```

Блоки кода

- Блоки кода обрамляются в фигурные скобки “{“ “}”
- Охватывают определение класса
- Определения методов
- Логически связанные разделы кода

```
import java.util.Date;
public class FirstProgram {
    public static void main (String[] args){
        System.out.println (new Date());
    }
}
```

Объявление методов

- Методы определяются только внутри класса
- Указывается
  - Модификатор доступа
  - Слово static
  - Тип возвращаемого значения
  - Аргументы

Пример

```
import java.util.Date;
public class FirstProgram {
    private Date today;
    public Date getToday(){
        return today;
    }
    public void setToday(Date aToday){
        int i = 0;
        i++
        today = aToday;
    }
}
```

Переменные

- Основное место для хранения данных
- Должны быть явно объявлены
- Каждая переменная имеет тип, идентификатор и область видимости
- Определяются для класса, для экземпляра и внутри метода

Объявление переменных

- Может быть объявлена в любом месте блока кода
- Должна быть объявлена перед использованием
- Обычно переменные объявляются в начале блока
- Область видимости определяется блоком
- Необходимо инициализировать переменные перед использованием
- Переменные простых типов инициализируются автоматически

Объявление переменных

- Основная форма объявления
  - тип идентификатор [= значение];

```
public class FirstProgram {
    public static void main (String[] args){
        int itemsSold = 10;
        float itemCost = 11.0f;
        int i, j, k;
        double interestRate;
    }
}
```

- При объявлении переменные могут быть проинициализированы

Именование переменных

- Имя переменной должно начинаться с буквы, знака подчеркивания или со знака “$”
- Имя переменной может включать цифры
- Давайте переменным осмысленные имена

Простые типы данных

- Восемь простых типов данных
  - Шесть числовых
  - Символьный
  - Логический
- Определяемые пользователем типы
  - Классы
  - Интерфейсы
  - Массивы

Простые типы данных

Целые	С плавающей точкой	Символьный	Логический
byte short int long	float double	char	boolean
1, 2, 3, -2 012 0x23f 2553L	3.0F .9937F 3.455E8 1.0D	‘s’ ‘\141’ ‘\u0061’ ‘\n’	true false

Операторы

- Пять типов операторов
  - Присваивание
  - Арифметические
  - Побитовый сдвиг
  - Равенство
  - Логические

Оператор присваивания

- Оператор присваивания – выражение и может использоваться там, где допустимы выражения
- Сначала вычисляется правая часть, а затем полученное значение присваивается левой части

```
int itemsSold = 10;
double itemCost = 11.0F+12.0D;
int i = i+7;
i = j = k = 100;
```

Арифметические операторы

- Сложение (+)
- Умножение (*)
- Вычитание (-)
- Деление (/)
- Остаток от деления (%)

Все арифметические операции производятся над int или long

ВНИМАНИЕ:

byte a = 100;

byte b = 100;

byte c = a+b;

c = -56!!!

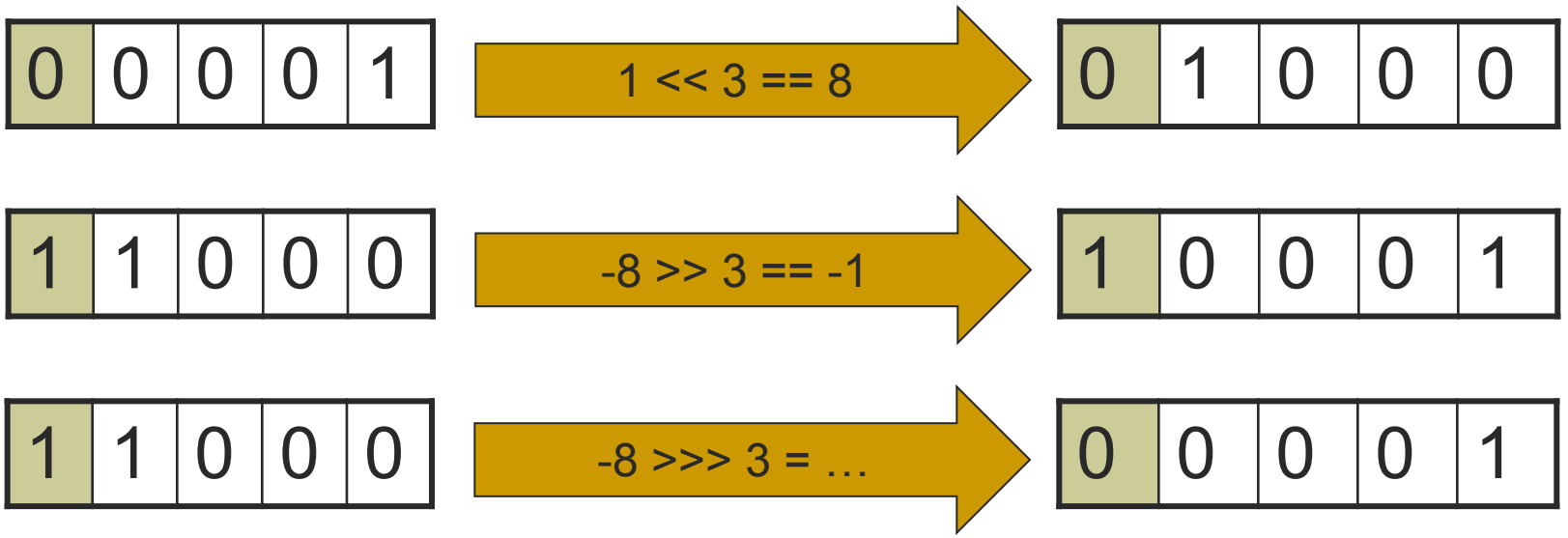
Операции инкремента и декремента

- Увеличение на 1 (++)
- Уменьшение на 1 (--)

```
int var1 = 3;
int var2 = 0;
var2 = ++var1; //сначала увеличивается var1, а затем
  //присваивается var2
var2 = var1++; //сначала присваивается var2, а затем
  // увеличивается var1
```

Побитовый сдвиг

- << - сдвиг влево
- >> - сдвиг вправо
- >>> - сдвиг вправо с заполнением нулями
- Правая часть сокращается до остатка от деления на длину числа, т.е. 1 << 35 == 1 << 3



Операторы сравнения

- < - меньше
- > - больше
- >= - больше или равно
- <= - меньше или равно
- == - равно
- != - не равно

Логические операторы

- && - and
- || - or
- ^ - xor
- ! – not

Приоритет операций

№	Операция	Порядок выполнения
1	[ ] . () (вызов метода)	Слева направо
2	! ~ ++ -- +(унарный) -(унарный) () (приведение) new	Справа налево
3	* / %	Слева направо
4	+ -	Слева направо
5	<< >> >>>	Слева направо
6	< <= > >= instanceof	Слева направо
7	== !=	Слева направо
8	&	Слева направо
9	^	Слева направо
10	|	Слева направо
11	&&	Слева направо
12	||	Слева направо
13	?:	Слева направо
14	= += -= *= /= %= |= ^= <<= >>= >>>=	Справа налево

Управление выполнением программы

- Типы выполнения
  - Последовательность
  - Выбор
  - Итерация
  - Переход

Последовательность

- Каждый оператор завершается точкой с запятой
- Группы операторов обрамляются фигурными скобками
- Каждая группа выполняется как единый оператор внутри последовательности операторов

Оператор `if`

```
if (логическое выражение)
    оператор1;
[else
    оператор2];
```

```
if (i % 2 == 0)
    System.out.println("Even");
else
    System.out.println("Odd");
```

```
if (i % 2 == 0){
    System.out.print(i);
    System.out.println(" is even");
}
```

Оператор **switch**

```
switch (выражение целого типа){
  case const1:
    statement1;
    break;
  case const2:
    statement2;
    break;
  default:
    statement3;
}
```

- Используется для выбора из счетного количества вариантов
- Выражения const должны быть типа byte, int, char или short

Циклы

- Три типа
  - while
  - do..while
  - for
- Все циклы имеют две части
  - Условие выполнения
  - Тело

Цикл **while**

```
while (логическое выражение)
    оператор;
```

```
int i = 0;
while (i < 100){
    System.out.println("i = "+i);
    i++;
}
```

Цикл do..while

```
do
    оператор;
while (условие выхода);
```

```
int i = 0;
do{
    System.out.println("i = "+i);
    i++;
}
while (i < 10);
```

Цикл for

```
for (инициализация; условие выхода; условие итерации)
    оператор;
```

```
for (int i = 0; i < 10; i++)
{
    System.out.println("i = "+i);
}
```

```
for (int i = 0, j = 10; i < j; i++, j--)
{
    System.out.println("i = "+i);
    System.out.println("j = "+j);
}
```

Переменная среды CLASSPATH

- Определяется в операционной системе
- Указывает JVM, где необходимо искать файлы .class
- Может ссылаться на каталоги и файлы .jar и .zip
- Интерпретатор загружает встроенные классы перед тем, как загрузить пользовательские
- Используется с командами **`java`** и **`javac`**

Выполнение JAVA программ

- Для того, чтобы класс можно было запустить, в нем должен быть определен метод **`main`**

```
public class FirstProgram {
    public static void main (String[] args){
        int itemsSold = 10;
        float itemCost = 11.0f;
        int i, j, k;
        double interestRate;
    }
}
```

Компиляция и запуск JAVA программ

- Компиляция .java файла

```
> javac [-classpath <path>] FirstProgram.java
```

- Запуск .class файла

```
> java [-classpath <path>] FirstProgram
```

Рекомендуемая литература

- **Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification**
- **Хорстманн, Корнелл Java2, в 2-х томах**
- **Bruce Eckel Thinking in Java**
- **http://java.sun.com/learning/tutorial/index.html**

Вопросы?