Вводный курс в Java
Занятие 2

Александр Русин
e-mail: alexander.rusin@simbirsoft.com

Android Developer
ООО СимбирСофт

Структура класса

```
//импорт пакетов
import java.util.*;
public class Sample {
    //метод класса
    public int method(int z) {
        return z;
    }
   //точка входа в программу
   public static void main(String[] args) {
       Sample s = new Sample();
       System.out.print(s.method(10));
  }
}
```

Элементы класса

```
public class Sample {
    private int x; // переменная экземпляра класса
    private int y = 0; // переменная экземпляра класса
    public final int CURRENT_YEAR = 2012; // константа
    protected static int bonus; // переменная класса
    static String version = "Java SE 7"; // переменная класса
    protected Calendar now;
   public int method(int z) {
        return z++;
   }
}
```

Модификаторы доступа

- private: члены класса доступны только внутри класса;
- default (package-private) (модификатор, по-умолчанию): члены класса видны внутри пакета (если класс будет так объявлен он будет доступен только внутри пакета);
- protected: члены класса доступны внутри пакета и в наследниках;
- public: члены класс доступны всем;

Модификаторы доступа

- static - ссылка этого поля у любого экземпляра класса будет ссылаться на одно и то же значение
- final – это модификатор, позволяющий объявлять константные поля в классе.

Конструкторы

```
public class Quest {
    // конструктор без параметров (по умолчанию)
    public Quest() {
        System.out.println("Вызван конструктор без параметров!!!");
    }
    // конструктор с параметрами
    public Quest(int idc, String txt) {
        super(); /* вызов конструктора супер класса явным образом
     необязателен,   компилятор вставит его автоматически*/
       System.out.println("Вызван конструктор с параметрами!!!");
       System.out.println(id + " " + text);
   }
}
```

Порядок инициализации класса

1. Статические поля
2. Поля
3. Конструктор

Порядок описания полей и методов не оказывает влияния на порядок инициализации.

Порядок инициализации класса

```
public class Department {
    { System.out.println("logic"); };                    //2
    static { System.out.println("static logic"); }    //1
    private int id = 7;
    public Department(int d) {
        id = d;
        System.out.println("конструктор");          //3
    }
    int getId() {  return id;   }
   { id = 10; System.out.println("logic"); }          //2
}
```

SimbirSoft software development

Наследование

Важнейший механизм ООП, позволяющий описать новый класс на основе уже существующего

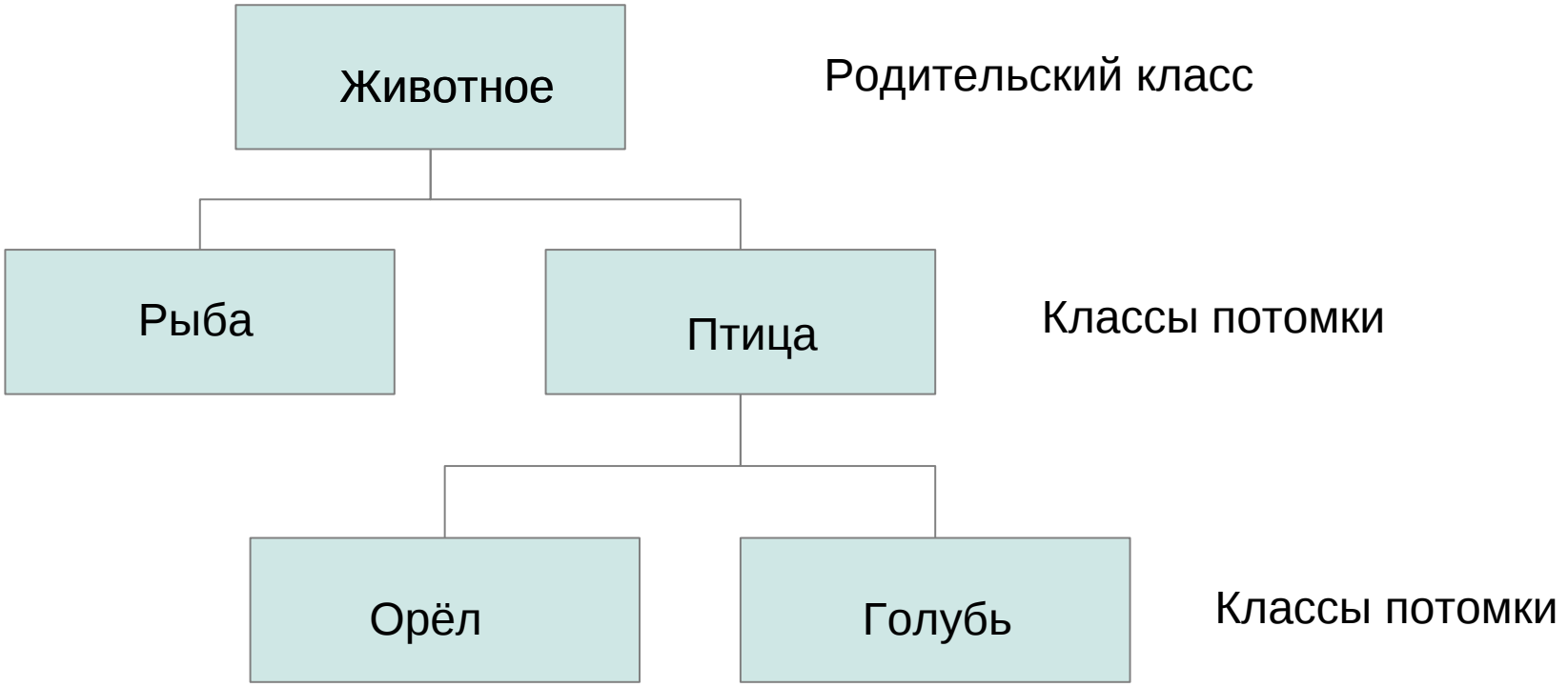
- При наследовании свойства и функциональность родительского класса наследуются новым классом
- Класс-наследник имеет доступ к публичным и защищённым методам и полям класса родительского класса
- Класс-наследник может добавлять свои данные и методы, а также переопределять методы базового класса

Терминология

- Родительский или базовый класс (класс-родитель) – класс, выступающий в качестве основы при наследовании
- Класс-потомок (дочерний класс, класс-наследник) – класс, образованный в результате наследования от родительского класса
- Иерархия наследования – отношения между родительским классом и его потомками
- Интерфейс класса – совокупность публичных методов класса, доступная для использования вне класса

  В интерфейсной части данные обычно не размещают
- Реализация класса – совокупность приватных методов и данных класса

Графическое изображение иерархий наследования



Перегрузка методов в классе наследнике

Методы производного класса замещает собой все методы родительского класса с тем же именем, количеством и типом аргументов.

SIMBIRSOFT
software development

Преимущества использования наследования

- Возможность создания новых типов, расширяя или используя функционал уже имеющихся
- Возможность существования нескольких реализаций одного и того же интерфейса
- Абстракция
- Полиморфизм

SimbirSoft software development

Инкапсуляция

- Понимается сокрытие информации о внутреннем устройстве объекта
- Работа с объектом может вестись только через его общедоступный (public) интерфейс.
- Другие объекты не должны вмешиваться в "дела" объекта, кроме как используя вызовы методов.

Инкапсуляция

```
public class Point {
    private final double x; //видимо только внутри класса
    private final double y;//видимо только внутри класса
    /** доступ к полю*/
    public double getX() {
        return x;
    }
    /** доступ к полю*/
    public double getY() {
        return y;
}
```

SIMBIRSOFT
software development

Абстракция

Абстра́кция (от лат. abstractio — «отвлечение») — отвлечение в процессе познания от несущественных сторон, свойств, связей предмета или явления с целью выделения их существенных, закономерных признаков.

Абстракция данных состоит в разделении несущественных деталей реализации подпрограммы и характеристик существенных для корректного её использования.

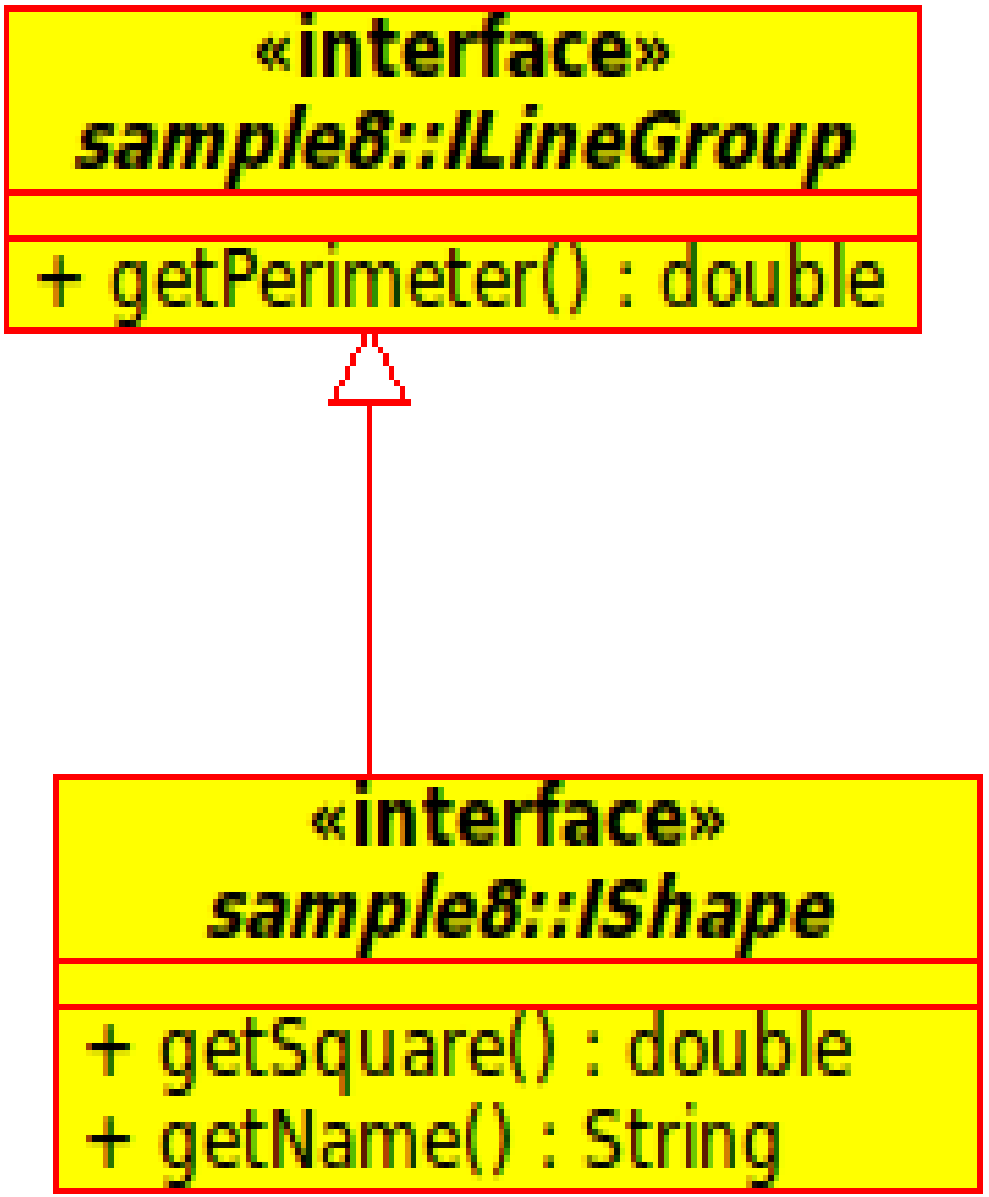
Абстрактный класс

```
public abstract class AbstractCourse {
    private String name;
    public AbstractCourse() {
    }
    public abstract void changeTeacher(int id);
   /*определение  метода отсутствует */
    public void setName(String n) {
        name = n;
    }
}
```

Интерфейс

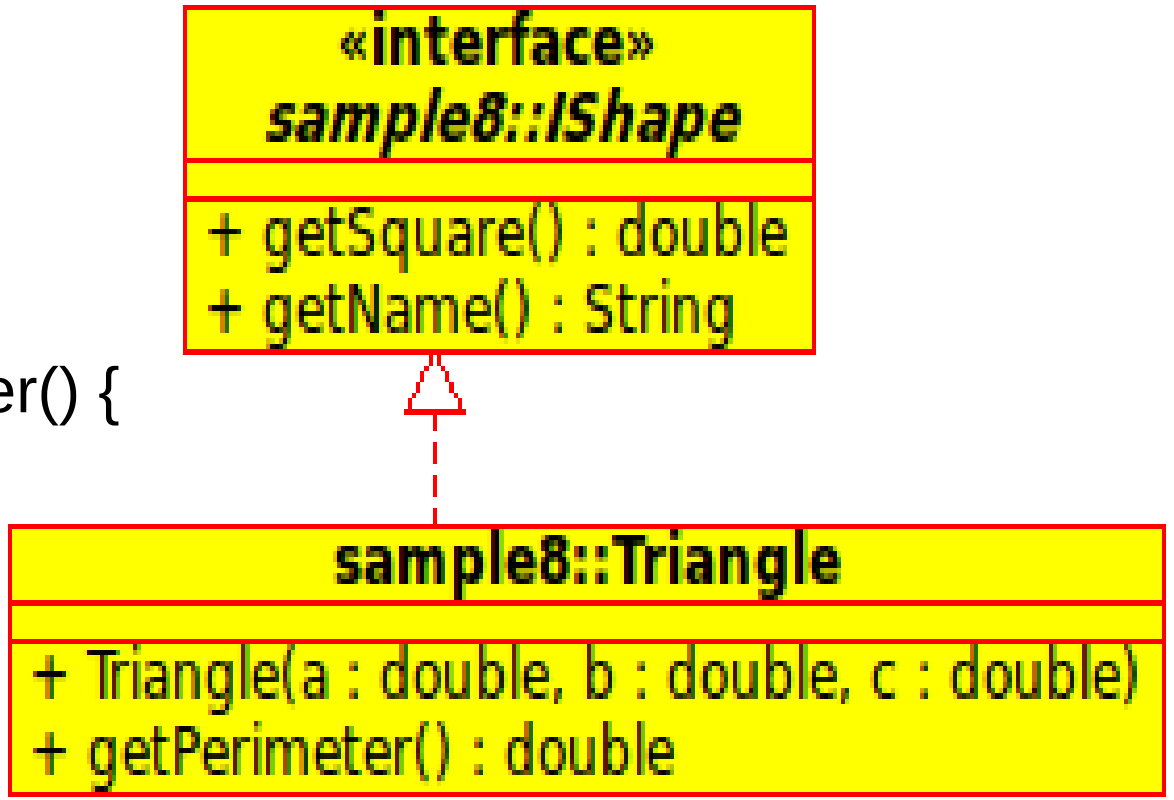
```
public interface IShape extends ILineGroup {
// int id; // ошибка, если нет инициализации
// void method(){} /* ошибка, так как абстрактный
   метод не может иметь тела!*/
   double getSquare(); // объявление метода
}
```

Интерфейс


«interface»
sample8::ILineGroup
+ getPerimeter() : double
«interface»
sample8::IShape
+ getSquare() : double
+ getName() : String

Пример

```
public abstract class Triangle implements IShape {
    private double a, b, c;
    public Triangle(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }
    public double getPerimeter() {
        return a + b + c;
    }
}
```



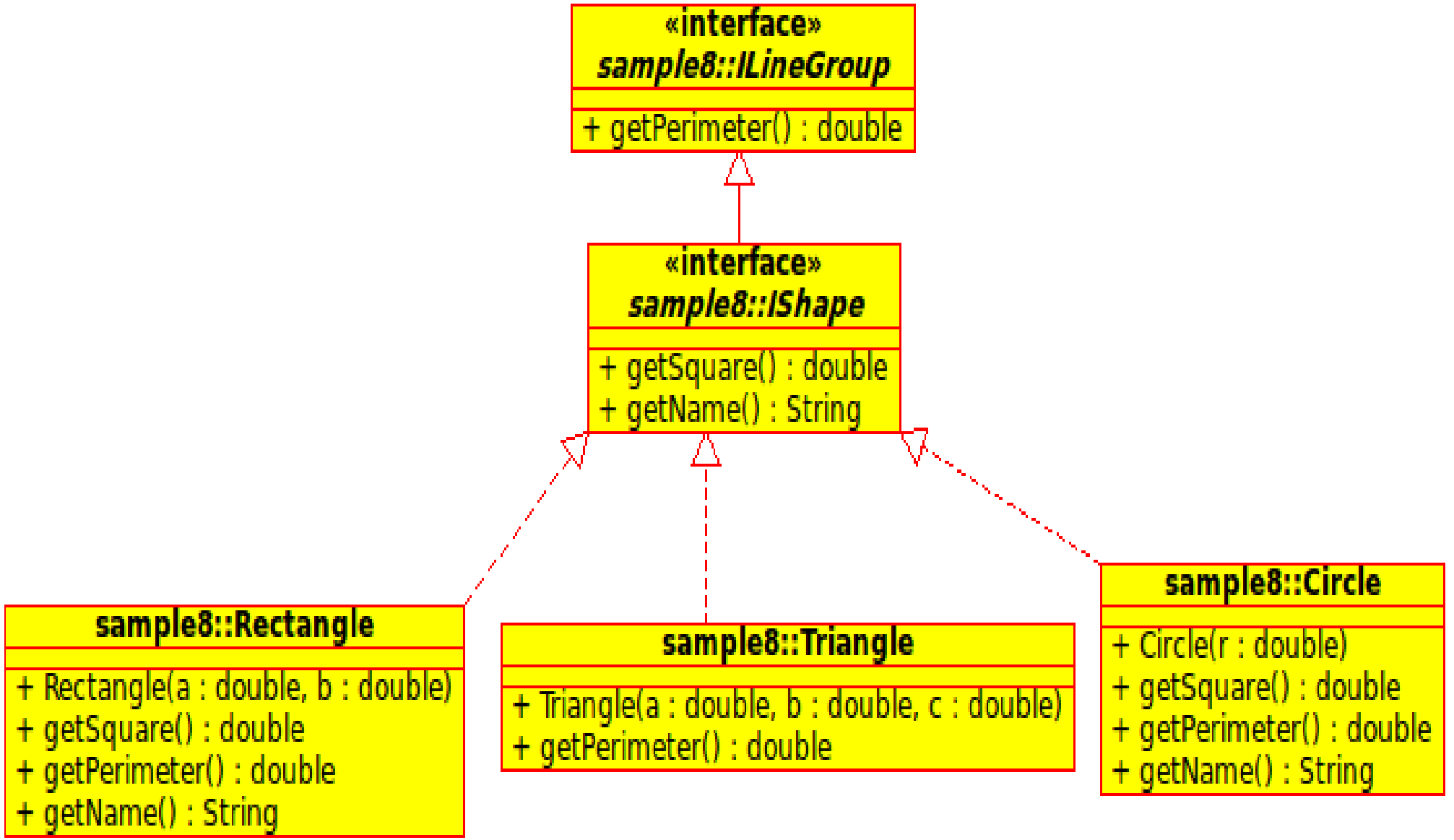
SIMBIRSOFT
software development

Полиморфизм

- Полиморфи́зм — возможность объектов с одинаковой спецификацией иметь различную реализацию.
- «Один интерфейс, множество реализаций»

SIMBIRSOFT
software development

Полиморфизм



Полиморфизм

```
public class Runner {
    public static void printFeatures(IShape f) {
        System.out.printf("название:%s площадь:%.2f периметр:
            %.2f%n",   f.getName(), f.getSquare(), f.getPerimeter());
    }
    public static void main(String[] args) {
        Rectangle r = new Rectangle(5, 9.95);
        Circle c = new Circle(7.01);
        printFeatures(r);
        printFeatures(c);
    }
}
```

Полиморфизм

Результат выполнения кода

- название:Прямоугольник площадь:49,75 периметр: 29,90
- название:Круг площадь:154,38 периметр: 44,05

SIMBIRSOFT
software development

Домашняя работа

1 Создать класс Pair (пара целых чисел); определить метод умножения на число и операцию сложения пар (a,b) + (c,d) = (a + b, c + d). Определить класс-наследник Money с полями: рубли и копейки. Переопределить операцию сложения и определить методы вычитания и деления денежных сумм.

2. Определить класс Pair с полями типа double. Реализовать операции сложения пар и умножения на число как в задании 1. Определить производный класс Complex и реализовать методы умножения (a, b) (c, d) = (ac - bd, ad + be) и вычитания (a, b) - (c, d) = (a - b, c - d).

SimbirSoft software development

Литература

- http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2 (определения)
- http://www.javable.com/tutorials/fesunov/lesson5/ (примеры)
- http://ru.wikibooks.org/wiki/Java
- http://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85 (Абстракция)

SIMBIRSOFT
software development

Спасибо за внимание!