

ПЛАТФОРМЕНО-НЕЗАВИСИМИ ПРОГРАМНИ ЕЗИЦИ

Лекция 7 НАСЛЕДСТВЕНОСТ

проф. дн И. Атанасов

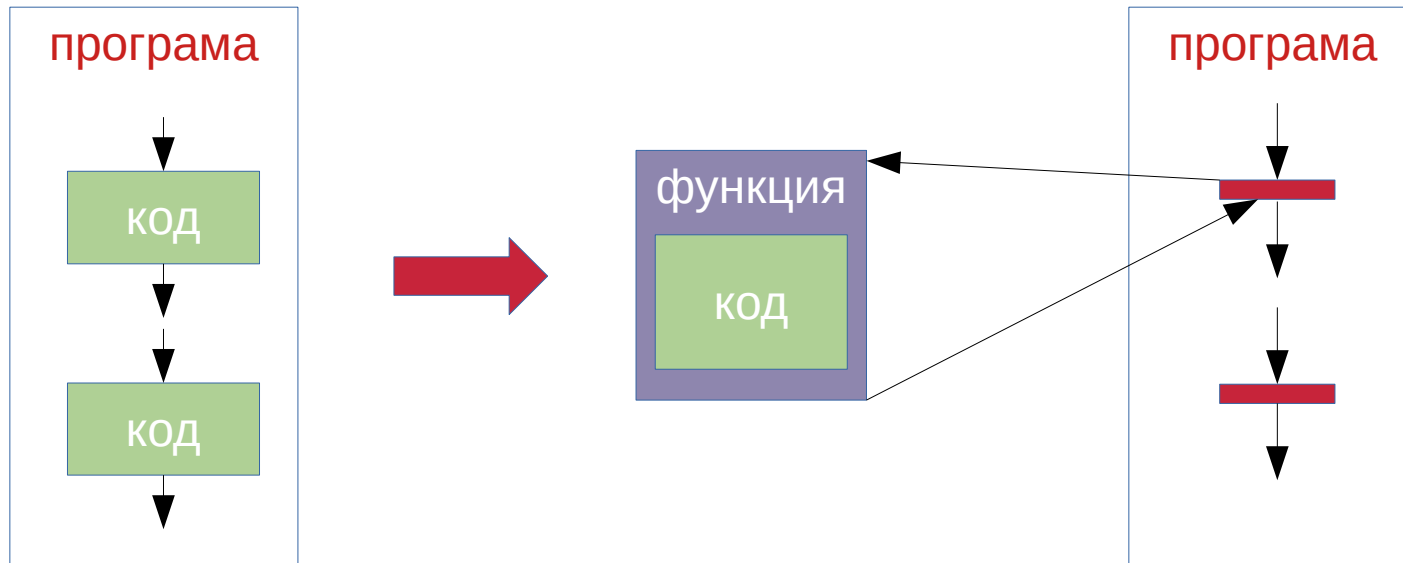
СЪДЪРЖАНИЕ

1. Постановка
2. Пример за наследственост
3. Интерпретация и основни понятия
4. Видове наследственост
5. Понятие за интерфейс
6. Предефиниране и полиморфизъм
7. Пример за предефиниране и полиморфизъм
8. Абстрактни методи и класове

НАСЛЕДСТВЕННОСТ

Постановка

Предпоставка: процедурно-ориентирано програмиране



НАСЛЕДСТВЕНОСТ

Пример за наследственост

Принцип: преизползване

```
class Shape {  
    boolean selected;  
  
    /*  
     *  
     */  
}
```

```
class Circle {  
    boolean selected;  
    double radius = 1.0;  
  
    /* ... */  
}
```

Има ли общо между двата класа?

НАСЛЕДСТВЕНОСТ

Пример за наследственост

Принцип: преизползване

```
class Shape {  
    boolean selected;  
  
    /*  
     *  
     */  
}
```

```
class Circle extends Shape {  
    double radius = 1.0;  
  
    /* ... */  
}
```

Има ли изменение? Какви са, ако има?

НАСЛЕДСТВЕНОСТ

Интерпретация

Наследственост:

Връзка между понятия от тип 'е' ('is-a')

От примера:

*Кръг е един вид Форма, който разширява основното понятие т.е. Форма, чрез добавяне на специфични свойства, които са наричани още полета или атрибути, а в конкретния случай това е *радиус*.*

НАСЛЕДСТВЕНОСТ

Основни понятия

Наследен клас:

Базов клас, Родителски клас, Родител

Наследяващ клас:

Наследник

От примера:

Форма е по-абстрактното понятие, което се специализира, конкретизира от понятието *Кръг* чрез наследяване.

НАСЛЕДСТВЕНОСТ

Видове наследственост

Единична наследственост:

Всеки *наследник* има **точно** един *Родителски клас*.

Множествена наследственост:

Всеки *наследник* има **поне** един *Родителски клас*.

Множествената наследственост може ли да предизвика проблемни ситуации?

НАСЛЕДСТВЕННОСТ

Наследственность при Java

```
class Circle extends Shape {  
    double radius = 1.0;  
    /* ... */  
}
```

```
class Circle extends Shape, Figure {  
    double radius = 1.0;  
    /* ... */  
}
```

Единична наследственность между класове!

НАСЛЕДСТВЕНОСТ

Понятие за интерфейс

Няколко метода (0, 1 или повече), които формират функционална група и имат смислова връзка помежду си.

```
public interface IDraw  
{  
    public void draw();  
}
```

Методите се задават само с тяхната сигнатура!

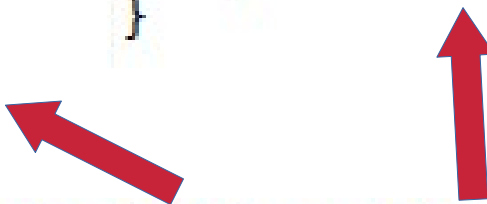
НАСЛЕДСТВЕННОСТЬ

Наследственность при интерфейсах

```
public interface IDraw
{
    public void draw();
}
```

```
public interface IMove
{
    public void move(int x, int y);
}
```

```
public interface IShape extends IDraw, IMove {
    /* ... */
}
```

Two red arrows originate from the 'extends IDraw, IMove' part of the IShape interface definition. One arrow points diagonally up and to the left towards the IDraw interface definition. The other arrow points diagonally up and to the right towards the IMove interface definition.


Множественная наследственность между интерфейсами!

НАСЛЕДСТВЕНОСТ


Предефиниране и полиморфизъм

```
public class Shape implements IDraw {  
    /* ... */  
    public void draw() {}  
    /* ... */  
}
```

```
public class Circle extends Shape {  
    /* ... */  
    public void draw() { /* .1. */ }  
    /* ... */  
}
```



```
public class Rectangle extends Shape {  
    /* ... */  
    public void draw() { /* .2. */ }  
    /* ... */  
}
```



НАСЛЕДСТВЕНОСТ

Предефиниране и полиморфизъм - пример

```
Circle c1 = new Circle();  
Rectangle r1 = new Rectangle();  
/*  
 *  
 */  
c1.draw();  
r1.draw();
```




```
Shape s1 = new Circle();  
Shape s2 = new Rectangle();  
/*  
 *  
 */  
s1.draw();  
s2.draw();
```

Има ли грешка?

Чий метод се извиква – на Shape или
на Rectangle?

НАСЛЕДСТВЕНОСТ

Абстрактни методи и класове

```
public abstract class Figure {  
    /**   
    *  
    */  
  
    public abstract void draw();  
}
```

За абстрактен метод трябва да се обяви всеки метод, който няма тяло, а клас, който има поне един абстрактен метод е също абстрактен.

НАСЛЕДСТВЕНОСТ

Абстрактни методи и класове

```
public abstract class Figure {  
    /**   
    *  
    */  
  
    public void draw() {  
        /* */  
    }  
}
```

За абстрактен клас може да се обяви и такъв клас, който няма нито един абстрактен метод. **Каква е ползата?**

НАСЛЕДСТВЕННОСТ

Абстрактни методи и класове

За абстрактен клас трябва да се обяви и този клас, който “обещава” да реализира даден интерфейс, но поне един от методите на интерфейса така и не се реализират.

```
public interface IDraw
{
    public void draw();
}
```

```
public abstract class Figure implements IDraw {
    /**
     *
     */
}
```


ВЪПРОСИ