

# Обекти в Java

Ненко Табаков, Пламен Танов, Любомир Чорбаджиев

Технологично училище “Електронни системи”  
Технически университет, София

1 април 2009



**Забележка:** Тази лекция е адаптация на:

- [Corey McCaffrey: \*Java Objects\*](#) from [6.092: Java for 6.170](#) (MIT OpenCourseWare: Massachusetts Institute of Technology)

**Лиценз:** [Creative commons BY-NC-SA](#)



Тези материали са разработени в рамките на проекта “Софтуерна академия “Електронни системи”, съфинансиран от Европейския съюз и Европейския социален фонд.

# Съдържание

# Препратки

- Когато в Java дефинирате променлива от даден клас, то всъщност дефинирате препратка (reference). Дефинирането на променлива не създава обект.
- Препратката или сочи към обект или е **null**.

```
1 Integer i1 = null;  
2 Integer i2 = new Integer(3);
```

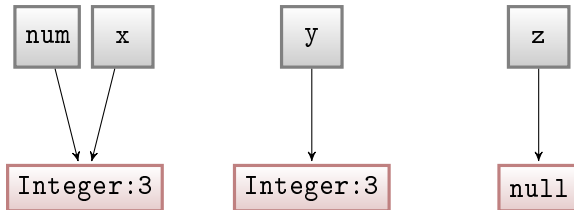
# Обекти

- Обектът е инстанция (екземпляр) на даден клас.
- За да се създаде обект е необходимо да се използва оператора **new**. Този оператор извиква конструктор на обекта.

```
1 Integer i1 = null;  
2 Integer i2 = new Integer(3);
```

# Пример

```
1 public class AssignmentReview {  
2     public static void main(String[] args) {  
3         Integer num=null;  
4         num = new Integer(3);  
5         Integer x = num;  
6         Integer y = new Integer(3);  
7         Integer z;  
8     }  
9 }
```



# Препратки `null`

- Ненасочените препратки сочат към `null`
- `null` не е обект (няма полета, нито методи)

```
1 Integer z;  
2 z.intValue(); //този ред ще генерира NullPointerException
```

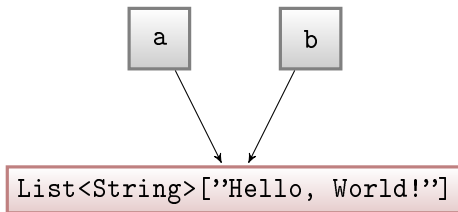
# Обекти и препратки

- Операторът за присвояване (=) насочва препратката към обект
- Методите са в състояние да променят вътрешното състояние на обекта
- Може да имате няколко препратки към един и същ обект, така че се пазете от странични ефекти



# Обекти и препратки

```
1 public class MutationExample {  
2     public static void main(String[] args) {  
3         List<String> a = new ArrayList<String>();  
4         List<String> b = a;  
5         a.add("Hello, world!");  
6         System.out.println(b);  
7     }  
8 }
```



# Статични и нестатични членове

- Методите и полетата могат да се декларират като статични
- Статичните методи/полета принадлежат на класа
- Нестатичните методи/полета принадлежат на обекта

# Пример: Нестатично поле

```
1 public class Bean {
2     public int beanCounter = 0;
3
4     public Bean() {
5         beanCounter++;
6     }
7
8     public static void main(String[] args) {
9         new Bean();
10        new Bean();
11        Bean bean = new Bean();
12        System.out.println(bean.beanCounter);
13        //prints 1
14    }
15 }
```

# Пример: Статично поле

```
1 public class Bean {
2     public static int beanCounter = 0;
3
4     public Bean() {
5         beanCounter++;
6     }
7
8     public static void main(String[] args) {
9         new Bean();
10        new Bean();
11        Bean bean = new Bean();
12        System.out.println(bean.beanCounter);
13        //prints 3
14    }
15 }
```

# Пример: Нестатичен метод

```
1 public class Bean {  
2     private boolean planted = false;  
3  
4     public void plantBean() {  
5         bean.planted = true;  
6     }  
7  
8     public static void main(String[] args) {  
9         Bean bean = new Bean();  
10        bean.plantBean(); // Invoked on instance  
11    }  
12 }
```

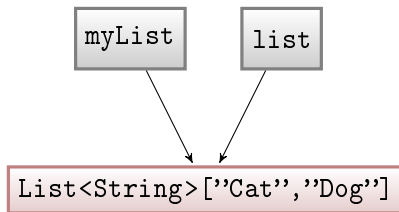
# Пример: Статичен метод

```
1 public class Bean {  
2     private boolean planted = false;  
3  
4     public static void plantBean(Bean bean) {  
5         bean.planted = true;  
6     }  
7  
8     public static void main(String[] args) {  
9         Bean bean = new Bean();  
10        Bean.plantBean(bean); // Invoked on class  
11        bean.plantBean(bean); // Legal but inadvisable!  
12    }  
13 }
```

## Пример: Обекти, предавани с препратки

```
1 public static <T> void removeFirst(List<T> list) {  
2     list.remove(0);  
3 }  
4  
5 public static void main(String[] args) {  
6     List<String> myList= new ArrayList<String>();  
7     myList.add("Cat");  
8     myList.add("Dog");  
9     System.out.println(myList); // Prints [Cat,Dog]  
10    removeFirst(myList);  
11    System.out.println(myList); // Prints [Dog]  
12 }
```

# Обекти и препратки





# Блокове и видимост

- Блокът (`{ }`) определя областта на видимост на препратките
- Препратките съществуват от момента на създаването им до момента, в който излязат от областта на видимост
- Полетата могат да се използват в рамките на класа
- Параметрите на даден метод могат да се използват в рамките на метода

# Пример: Област на видимост

```
1 public class ScopeExample {  
2     private int field;  
3  
4     public int method(int parameter) {  
5         int localVar1;  
6         if (field > 0) {  
7             int x;  
8         }  
9         int localVar2;  
10    }  
11 }
```

# Пример: Област на видимост

```
1 public class ScopeExample {  
2     private int field;  
3  
4     public int method(int parameter) {  
5         int field; // Legal, but hides field!  
6         int localVar;  
7         if (this.field > 0) { // Accesses field  
8             int x;  
9         }  
10        int localVar; // Illegal: same scope  
11    }  
12 }
```

# Изводи

- Препратките сочат към обекти. Трябва да се прави разлика между препратка и обект.
- Пазете се от препратки, които са **null**
- Не извиквайте статични методи чрез инстанции
- Ако не искате даден обект да се променя, то когато го предавате като аргумент трябва изрично да направите копие.
- Минимизирайте видимостта на променливите