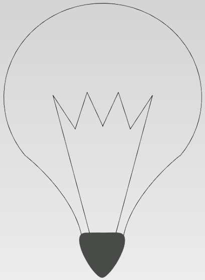
Using objects

Adam Krechowicz

Structural programming

- Data
- Subprograms

Bruce Eckel's Bulb



Object oriented programming

- encapsulation
- hierarchy
- composition
- re-usability

Class

Class

A definition of a being. It represent the way of using, performing and structure of modeled element.

Class consists of

- Fields
- Methods

Example class

```
package com.wyklad.objects;

public class Bulb {

  int brightness;

  public Bulb(int b) {
    super();
    this.brightness = b;
  }

  public void on(){

  }

  public void off(){

  }

}
```

Object

Object

Instance of a class. An object is created based on a specified class.

Example of using object

```
public static void main(String[] args){
  Bulb bulb = new Bulb(100);
  bulb.on();
  bulb.off();
}
```

Java

- Object oriented language
- "Everything is an object"

Primitive data types

type	name	wrapping class
logic value	boolean	Boolean
char	char	Character
8 bits digit	byte	Byte
16 bits digit	short	Short
32 bits digit	int	Integer
64 bits digit	long	Long
32 bits floating point digit	float	Float
64 bits floating point digit	double	Double

Range

Negative numbers

All digit data types are considered signed

Using of primitive types

```
package com.wyklad.objects;

public class Obiekt {

  int pole = 4;
  long pole2 = 123L;

  public Obiekt() {
    super();
  }

  public static void main(String[] args){
    byte a = 3;
    char c = 4;
    System.out.println(a);
    short z = 5;
  }

}
```

Integer operations

- assigning:

```
int a = 3;
```

- basic arithmetic operations:

```
a = 3 + 1;
a = 3 - 1;
a = 3 * 2;
a = 4 / 2;
a = 3 % 2;
```

- short arithmetic operations:

```
a++; ++a;
a--; --a;
a+= 2;
a-= 2;
```

Bitwise operations

- negation:

```
a = ~a;
```

- bit shifting:

```
a << 1;
a >> 1;
a >>> 1;
```

- logic operations:

```
a & b;
a | b;
a ^ b;
```

Basic operations

```
if (a == 1){
..
}
while (c == true){
..
}
do{
..
} while (c == false)
```

Initialization

```java
public static void main(String[] args){
  int a = 5;
  int b;
  System.out.println(a);
  System.out.println(b);
  int c;
  if (a > 5)
    c = 1;
  System.out.println(c);
}
```

Methods

```
int method1(){
  return 2;
}

int method2(int parameter){
  return parameter*2;
}

int method3(){
  System.out.println("Hello");
  return 5;
  System.out.println("World");
}
```

Methods

Not returning anything

If method do not return anything we use *void*

```
void method4(){
  System.out.println("Hello World");
}
```

Object initialization

```
public static void main(String[] args) {
  Bulb bulb = new Bulb(10);
  Bulb a;
  a = new Bulb(20);
  Bulb b = null;
}
```

Constructors

```
package com.wyklad.objects;

public class Constructor {

  public Constructor() {
    super();
  }

  public Constructor(int param){
    super();
  }
}
```

Methods using

Methods

Methods are executed on an objects

```
package com.wyklad.objects;

public class Program {

  void method(){

  }

  public static void main(String[] args){
    //method();
    Program program = new Program();
    program.method();
  }
}
```

Static elements

```
public class Statycznie {

  static int value = 5;

  static void method(){

  }

  public static void main(String[] args) {
    Statycznie a = new Statycznie();
    Statycznie b = new Statycznie();
    System.out.println(a.value);
    System.out.println(b.value);
    a.value++;
    System.out.println(a.value);
    System.out.println(b.value);
    a.method();
    b.method();
    method();
  }
}
```

Integer class

- parsing
- converting between different system numbers
- minimal and maximal value

Autoboxing

```java
package com.wyklad.objects;

public class Autoboxing {

  public Autoboxing() {
    super();
  }

  public static void main(String[] args){

    Integer i = new Integer(321);
    System.out.println(i);
    Integer j = 123;
    String s = j.toString();
    System.out.println(s);
  }
}
```

Unboxing

```java
package com.wyklad.objects;

public class Unboxing {

  public Unboxing() {
    super();
  }

  void method(int x){
    System.out.println(x);
  }

  public static void main(String[] args){
    Integer i = new Integer(123);
    int j = i;
    System.out.println(j);
    new Unboxing().method(i);
  }
}
```

BigInteger

```java
long l = Long.MAX_VALUE;
long k = l+l;
System.out.println("l+l = "+k);
BigInteger bi = new BigInteger(String.valueOf(l));
bi = bi.add(bi);
System.out.println("l+l = "+bi)
```

Results

```
l+l = -2
l+l = 18446744073709551614
```

BigDecimal

```java
public static void main(String[] args){
  double d = 0.1;
  BigDecimal bd = new BigDecimal(d);
  System.out.println("bd = "+bd);
  BigDecimal bd1 = new BigDecimal("0.1");
  System.out.println("bd1 = "+bd1);
  System.out.println("bd+bd1= "+bd1.add(bd));
}
```

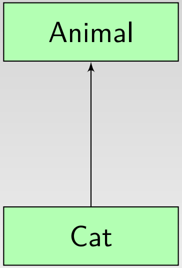
Results

```
bd = 0.1000000000000000055511151231257827021181583404541015625
bd1 = 0.1
bd+bd1= 0.2000000000000000055511151231257827021181583404541015625
```

String class

- Represents a string literal
- Operations:
  - Finding a substring
  - Concatenation
  - Replacing some content
  - Cutting a substring

Inheritance



Polymorphism

- Methods overriding
- Methods overloading

Object class

- clone()
- equals()
- hashCode()
- finalize()
- toString()
- getClass()
- wait(), notify(), notifyAll()

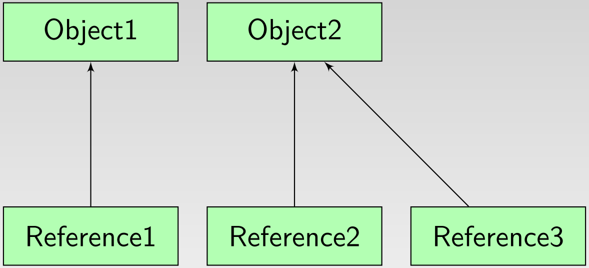
Comparison

```
package com.wyklad.objects;

public class Equals {
  public Equals() {
    super();
  }

  public static void main(String[] args){
    Bulb a = new Bulb(10);
    Bulb b = new Bulb(20);
    Bulb c = new Bulb(10);
    System.out.println(a == a);
    System.out.println(a.equals(a));
    System.out.println(a == b);
    System.out.println(a.equals(b));
    System.out.println(a.equals(c));
  }
}
```

Referencje

```
package com.wyklad.objects;

public class Reference {

  public Reference() {
    super();
  }

  public static void main(String[] args){
    Reference a = new Reference();
    Reference b = a;
    System.out.println(a == b);
    System.out.println(a.equals(b));
  }

}
```

Referencje



this

```java
package com.wyklad.objects;

public class This {

  int a;
  int b;

  void metoda(int b){
    a = b;
    this.a = b;
    this.b = b;
  }

}
```

String literals equality

```
package com.wyklad.objects;

public class StringEquals {

  public StringEquals() {
    super();
  }

  public static void main(String[] args){
    String a = new String("Hello World");
    String b = new String("Ala ma koda");
    String c = new String("Hello World");
    System.out.println(a == b);
    System.out.println(a.equals(b));
    System.out.println(a == c);
    System.out.println(a.equals(c));
  }

}
```

Metoda haszująca

```
package com.wyklad.objects;

  public static void main(String[] args) {
    Hash a = new Hash();
    Hash b = new Hash();
    String s = new String("Hello World");
    String s2 = new String("Hello World");
    String s3 = "Hello World";
    String s4 = "Hello World";
    System.out.println(a.hashCode());
    System.out.println(b.hashCode());
    System.out.println(s.hashCode());
    System.out.println(s2.hashCode());
    System.out.println(s3.hashCode());
    System.out.println(s4.hashCode());
  }
}
```

toString()

```java
public class ToString {
  public static void main(String[] args) {
    ToString a = new ToString();
    System.out.println(a);
    System.out.println(Integer.toHexString(a.hashCode()));
  }
}
```

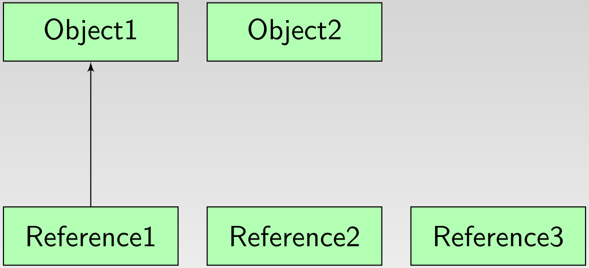
Rezultat

```
ToString@18a992f
18a992f
```

Object lifetime

```
package com.wyklad.objects;

public class Klasa {

  public Klasa() {
    super();
  }

  public static void main(String[] args){
    Klasa k = new Klasa();
    k = new Klasa();
    k = new Klasa();
    k = null;
  }
}
```

Referencje



Garbage Collector

```
package com.wyklad.objects;

public class Klasa {

  public Klasa() {
    super();
  }

  public static void main(String[] args){
    Klasa k = new Klasa();
    k = new Klasa();
    k = new Klasa();
    System.gc();
  }
}
```

Zakresy

```java
package com.wyklad.objects;

public class Scope {

  public static void main(String[] args){
    int a = 3;
    {
     int b = 5;
     Scope s = new Scope();
     System.out.println(a);
     System.out.println(b);
     System.out.println(s);
    }
    System.out.println(a);
    //System.out. println (b);
    //System.out. println (s);
  }

}
```

Naming conventions

- class names starts with capital letter
- object names starts with lowercase letter
- method names starts with lowercase
- constructor name must be the same as class name
- package names starts with lowercase
- if name consists of a couple of words we use *camelization* np. longNameWithManyWords
- constant names are composed with all uppercase letters ("_" might be used to separate words)

Access methods

```
package com.wyklad.objects;

public class Access {

  int value;

  int getValue(){
    return value;
  }

  void setValue(int value){
    this.value = value;
  }

  public static void main(String[] args) {
    Access access = new Access();
    access.setValue(123);
    System.out.println(access.getValue());
  }

}
```

JavaBeans

- public default constructor (without parameters)
- all fields are private and are accessed via access methods (set.., get..)
- serializable

JavaBeans – example

```
package com.wyklad.objects;

import java.io.Serializable;

public class Bean implements Serializable {

  private int key;
  private String value;

  public Bean() {
    super();
  }

  public void setKey(int key) {this.key = key;}

  public int getKey() {return key;}

  public void setValue(String value) {this.value = value;}

  public String getValue() {return value;}
}
```