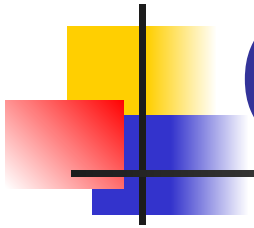
Reflection

Поняття про відображення (reflection, інтроспекція)



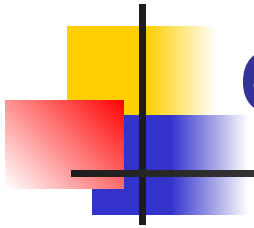
- Засоби, які дозволяють під час виконання програми отримувати інформацію про класи, їх поля і методи
- Більш загально – для динамічної роботи з кодом. Наприклад – створити екземпляр класу, назва якого вводиться в рядку.
- Базовий інструмент – **клас Class**.
- Додаткові можливості – пакет **java.lang.reflect**.

Отримання інформації про клас – базові можливості

```
String s = "qwerty";
        Class kl = s.getClass();
        System.out.println
    (kl.getSimpleName() + " is a
    subclass of " +

    kl.getSuperclass().getSimpleName())
    ;
```

Три способи отримання екземпляру класу Class

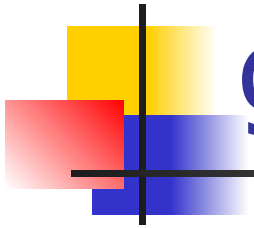


- метод ***getClass()*** класу Object;
- виклик статичного методу ***Class.forName***(ім'я класу) – завантаження класу за іменем;
- ***T.class***, де T-деякий тип.

Приклад: отримання методів класу

```
import java.io.*;
import java.lang.reflect.*;
public class reflection {
public static void main(String[] args) throws Exception {
BufferedReader br=new BufferedReader(new InputStreamReader
(System.in));
System.out.println("Enter class name");
String clName=br.readLine();
Class cl = Class.forName(clName);
 Method[] M  = cl.getMethods();
for (Method m:M) {System.out.println(m);}
}
}
```

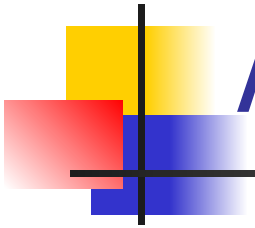
Різниця між getMethods та getDeclaredMethods



- Перший повертає всі публічні методи з урахуванням успадкування, а другий – всі методи, визначені в цьому класі.
- Задача: вивести список всіх методів, які визначені в класі, але лише публічних.

Динамічне завантаження класу та створення екземпляру

```
public static void main(String[] args) throws Exception {
        Class c = Class.forName("dinload.A");
        A a = (A) c.newInstance();
        a.inform();
    }
}
class A {
    static {        System.out.println("A is initialized");    }
    A() {        System.out.println("Instance created");    }
    void inform() { System.out.println(" You can create
    dinamic instance");    }
}
```

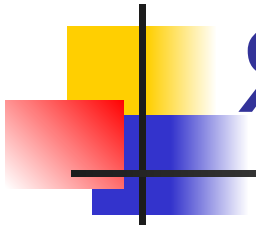


Або:

Class <Kl> cl = Kl.class;

Kl ekz = cl.newInstance();

ekz.metod();



Як це зробити більш гнучко?

- Треба уникнути зведень до класу, який за ідеєю прикладу має бути невідомим.
- Крім того, і ім’я методу може бути невідомим на етапі компіляції.
- Як це зробити?

Злам приватності: клас

```
class Bastion {
private String msg = "You cannot change this string";

public String getMsg() {
        return msg;
    }

    private void metod() {
        System.out.println("This method is private!");
    }
}
```

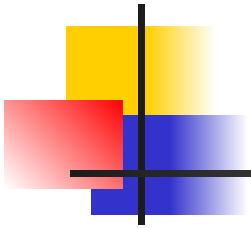
Злам приватності: власне злам

```
Bastion b = new Bastion();
        System.out.println("Our field is: "+b.getMsg());
        //b.str="Ku-ku"; - це не компілюється
        Class bk = b.getClass();
        Field fld = bk.getDeclaredField("msg");
        fld.setAccessible(true);
       fld.set(b, "Das ist ku-ku");
       fld.setAccessible(false);
        System.out.println("The cracked field is: "+b.getMsg());
        //b.inform(); - це не компілюється
        Method m = bk.getDeclaredMethod("metod");
        m.setAccessible(true);
       m.invoke(b);
        m.setAccessible(false);
```

Вказівники на функцію: клас з функцією…

```
class Ext {
private int Pole;

    public Ext(int Pole) {
        this.Pole = Pole;
    }
public void metod() {
    System.out.println("The field of this object is
    "+Pole);
}
}
```



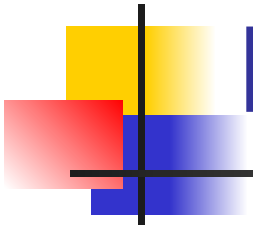
… і виклик функції

```
Method m =
  Ext.class.getMethod("metod");
      Ext ext = new Ext(50);
      m.invoke(ext);
```

Ще один приклад: виведення таблиці значень функції

```
Method square = Ext.class.getMethod("square",
   double.class);
      Method sqrt = Math.class.getMethod("sqrt",
   double.class);
…
 for (double x=from; x<=to; x+=dx) {

          double y = (Double) f.invoke(null, x);
          System.out.printf("%10.4f | %10.4f%n",x,y);
 }
```



Конструктивна приватність

Вправа: Написати консольне застосування, яке за допомогою Reflection (виклик newInstance) створює екземпляр класу, конструктор якого описаний як private.