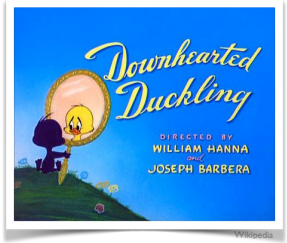
Programare Orientată pe Obiecte

Reflexia în Java

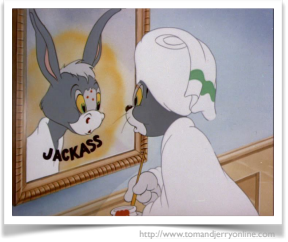
Dr. Petru Florin Mihancea

V20180924

Reflexia



Wikipedia



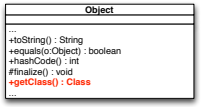
http://www.tomandjerryonline.com

... capacitatea unui program de a-și observa și gestiona la execuție propria structură

A

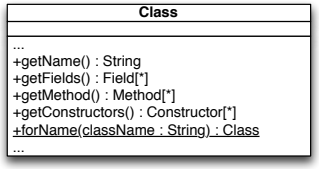
The Class class

Meta-clasa **Class**



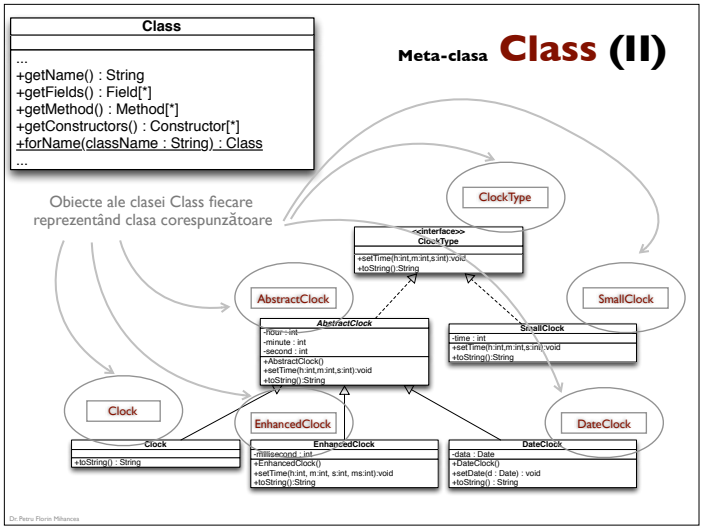
de exemplu, clasa Object este reprezentată la execuția unui program de un obiect instanță a acestei clase:

```
Class aCls =
  Class.forName("java.lang.Object");
System.out.println(aCls.getName());
```



un obiect al clasei Class reprezintă o clasă din cadrul programului

Meta-clasa Class (II)



ATENȚIE !!!

în programarea orientată pe obiecte noțiunea de clasă și obiect sunt evident diferite

în mecanismul de reflexie este doar o chestiune de reprezentare și nu se schimbă aceste noțiuni; o clasă din program este reprezentată de un obiect și, ca orice obiect, e definit de o clasă (meta-clasă) în speță clasa *Class*

Accesul la obiectele Class

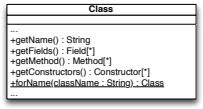
1) getClass():Class definită în Object

2) NumeClasa.class ex. Integer.class, int.class, Object.class, etc.

3) Pt. clasele înfășurătoare există un câmp special TYPE ex. Integer.TYPE

4) forName(nume:String):Class

Numele complet calificat



bytecode-ul clasei trebuie să fie găsit la runtime ex. în folderele specificate prin classpath altfel se aruncă excepția verificată ClassNotFoundException

Referințe la obiectele Class

```
Class c; //poate referii orice obiect class
c = Integer.class;
c = Object.class;
```

unele erori de programare pot fi observate târziu, la runtime; nu ne-ar putea ajuta compilatorul ?

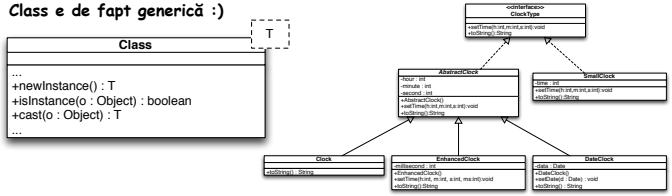
```
Class<Integer> i; //numai obiectul class reprezentând clasa Integer
i = Integer.class;
i = int.class;
i = Object.class; //Eroare de compilare

Class<Number> n; //numai obiectul class reprezentând clasa Number din biblioteca
                 //Java (superclasă pentru clasele înfășurătoare numerice)
n = Number.class;
n = Integer.class; //Eroare de compilare
n = Object.class; //Eroare de compilare

Class<? extends Number> nb; //obiectul class reprezentând clasa Number ori o
                            //subclasă a lui Number
nb = Number.class;
nb = Integer.class;
nb = Object.class; //Eroare de compilare
```

Metode interesante din Class

Class e de fapt generică :)



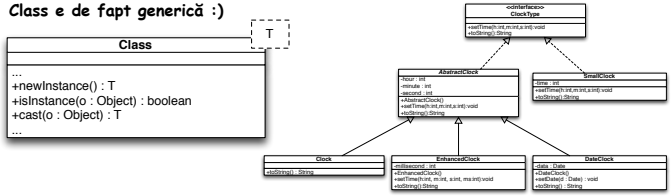

```
Class clockClass = Clock.class;
Clock aClock = (Clock) clockClass.newInstance();

Class<Clock> clockClassGeneric = Clock.class;
aClock = clockClassGeneric.newInstance();
```

crează o instanță a clasei respective :)
trebuie să aibă constructor no-arg altfel se arunca excepție verificată
(există variante și pentru cazul când avem numai constructori cu argumente)

Metode interesante din Class

Class e de fapt generică :)

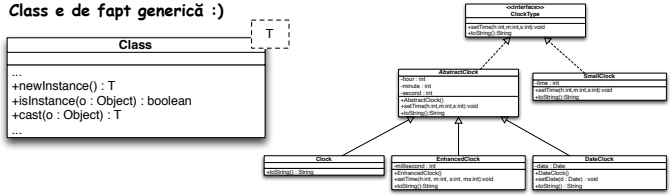



```
Class clockTypeClass = ClockType.class;
if(clockTypeClass.isInstance(someObjectRef)) {
  ...
}
```

true dacă obiectul referit de someObjectRef este de tipul reprezentat de obiectul referit de clockTypeClass (ca la operatorul instanceof)

Metode interesante din Class

Class e de fapt generică :)



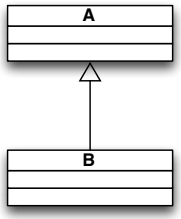

```
Class clockTypeClass = ClockType.class;
if(clockTypeClass.isInstance(someObject)) {
    ClockType aClock = (ClockType)clockTypeClass.cast(someObject);
    ...
}
sau folosind generics
Class<ClockType> clockTypeClass = ClockType.class;
if(clockTypeClass.isInstance(someObject)) {
    ClockType aClock = clockTypeClass.cast(someObject);
    ...
}
```

Putem schimba la runtime clasa instanţiată ori spre care se face verificarea isInstance (ceea ce nu se poate cu operatorul new ori instanceof)

Putem trece de unele limitări de la genericitatea din Java.

Realizează un cast

Quiz




```
Object x = new B();
```

```
x instanceof B
```

TRUE

```
x instanceof A
```

TRUE

```
x.getClass().equals(A.class);
```

FALSE

```
x.getClass() == A.class
```

FALSE

```
x.getClass().equals(B.class)
```

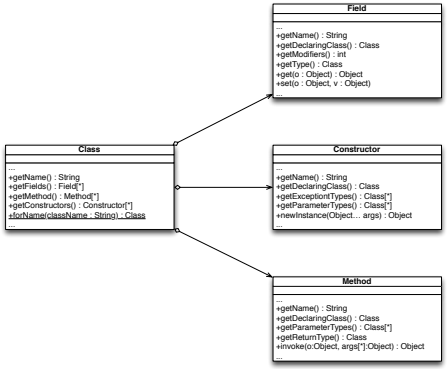
TRUE

B

The **Class** class is only the beginning :)

Meta-Modelul Java

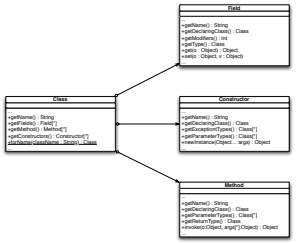
(conceptual și simplificat)



java.lang.reflect

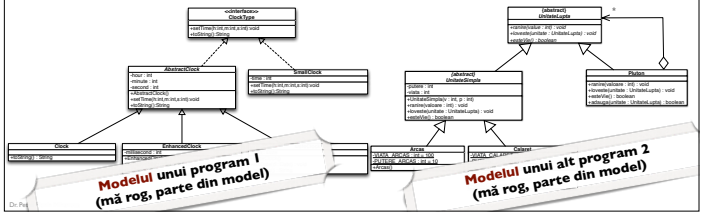
Meta-Modelul Java

(conceptual și simplificat)



Meta-Model - Un model al tuturor modelelor de programe

java.lang.reflect



Modelul unui program 1
(mă rog, parte din model)

Modelul unui alt program 2
(mă rog, parte din model)

Dr. Pet

De ce ?

în aceste exemple, folderele referite prin nume scurt sunt în directorul curent

Exemplul 1 - Oglinda :)

```
package exemple.oglinda;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
public class Mirror {
 public static void main(String[] args) {
  for(String aClassName : args) {
   try {
    Class aClass = Class.forName(aClassName);
    System.out.println(aClass.getName());
    System.out.println("\tIs an interface? :" + aClass.isInterface());
    for(Field aField : aClass.getFields()) {
     System.out.println("\tField name:" + aField.getName());
     System.out.println("\t\tField type:" + aField.getType().getName());
    }
    for(Method aMethod : aClass.getMethods()) {
     System.out.println("\tMethod name:" + aMethod.getName());
     System.out.println("\t\tMethod return type:" + aMethod.getReturnType().getName());
     for(int i = 0; i < aMethod.getParameterTypes().length; i++) {
      System.out.println("\t\t Arg " + i + ":" + aMethod.getParameterTypes()[i].getName());
     }
    }
    System.out.println();
   } catch(ClassNotFoundException e) {
     System.err.println("Class " + aClassName + " not found!");
   }
}}}
```

Exemplul I - Oglinda :)

```
package exemple.oglinda;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
public class Mirror {
 public static void main(String[] args) {
  for(String aClassName : args) {
   try {
    Class aClass = Class.forName(aClassName
    System.out.println(aClass.getName());
    System.out.println("\tIs an interface? :" + a
    for(Field aField : aClass.getFields()) {
     System.out.println("\tField name:" + aField
     System.out.println("\t\tField type:" + aFiel
    }
    for(Method aMethod : aClass.getMethods(
     System.out.println("\tMethod name:" + aM
     System.out.println("\t\tMethod return typ
     for(int i = 0; i < aMethod.getParameterTyp
      System.out.println("\t\t Arg " + i + ":" + a
     }
    }
    System.out.println();
   } catch(ClassNotFoundException e) {
     System.err.println("Class " + aClassName + " not found!");
   }
}}}
```

```
Pepis-MacBook-Pro:Desktop Pepi$ javac -d bin/ Mirror.java
Pepis-MacBook-Pro:Desktop Pepi$ java -cp bin exemple.oglinda.Mirror
Pepis-MacBook-Pro:Desktop Pepi$ java -cp bin exemple.oglinda.Mirror exemple.oglinda.Mirror
exemple.oglinda.Mirror
        Is an interface? :false
        Method name:main
                Method return type:void
                Arg 0:[Ljava.lang.String;
        Method name:wait
                Method return type:void
                Arg 0:long
                Arg 1:int
        Method name:wait
                Method return type:void
                Arg 0:long
        Method name:wait
                Method return type:void
        Method name:equals
                Method return type:boolean
                Arg 0:java.lang.Object
        Method name:toString
                Method return type:java.lang.String
        Method name:hashCode
                Method return type:int
        Method name:getClass
                Method return type:java.lang.Class
        Method name:notify
                Method return type:void
        Method name:notifyAll
                Method return type:void
```

Și pot face asta pentru orice clasă, specificând numele ei complet calificat; codul ei trebuie să fie găsit la rulare în classpath-ul dat

Exemplul 2 - Plugin-uri

```java
package exemple.plugins;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Main {
 public static void main(String[] args) throws IOException {
   BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
   String aPluginClassName;
   while((aPluginClassName = bf.readLine()) != null) {
      try {
        Class loadedClass = Class.forName(aPluginClassName);
        Class pluginClass = Plugin.class;
        Object anObject = loadedClass.newInstance();
         if(pluginClass.isInstance(anObject)) {
           Plugin aPlugin = (Plugin)anObject;
          aPlugin.execute();
         }
      } catch(ClassNotFoundException e) {
        System.out.println("Plugin class not found!");
      } catch(InstantiationException | IllegalAccessException e) {
        System.out.println("Plugin class cannot be instantiated - concrete/accesible/no-arg constructors?");
      }
   }
 }
}
```

```java
package exemple.plugins;
public interface Plugin {
      public void execute();
}
```

Exemplul 2 - Plugin-uri

```
package exemple.plugins;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Main {
 public static void main(String[] args) throws IOException {
   BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
   String aPluginClassName;
   while((aPluginClassName = bf.readLine()) != null) {
     try {
       Class loadedClass = Class.forName(aPluginClassName);
       Class pluginClass = Plugin.class;
       Object anObject = loadedClass.newInstance();
        if(pluginC
          Plugin a
          aPlugin.
        }
     } catch(Cla
       System.o
     } catch(Inst
       System.o                                                    2");
     }
   }
 }
}
```

```
package exemple.plugins;
public interface Plugin {
    public void execute();
}
```

```
Pepis-MacBook-Pro:Desktop Pepi$ javac -d bin Plugin.java Main.java
Pepis-MacBook-Pro:Desktop Pepi$ java -cp bin:/Users/Pepi/Desktop/dropplugins/
exemple.plugins.Main
test
Plugin class not found!
exemple.plugins.Plugin
Plugin class cannot be instantiated - concrete/accesible/no-arg constructors?
personal.MyPlugin
Plugin class not found!
```

Exemplul 2 - Plugin-uri (II)

```
package personal;

public class MyPlugin implements exemple.plugins.Plugin {
    public void execute() {
        System.out.println("Plugin-ul meu se executa!");
    }
}
```

```
javac -cp bin/ -d dropplugins/ MyPlugin.java
```

```
Pepis-MacBook-Pro:Desktop Pepi$ javac -d bin Plugin.java Main.java
Pepis-MacBook-Pro:Desktop Pepi$ java -cp bin:/Users/Pepi/Desktop/dropplugins/
exemple.plugins.Main
test
Plugin class not found!
exemple.plugins.Plugin
Plugin class cannot be instantiated - concrete/accesible/no-arg constructors?
personal.MyPlugin
Plugin class not found!
personal.MyPlugin
Plugin-ul meu se executa!
```

Reîncărcarea unei clase modificate nu e simplă :)

Exemplul 3 - Plugin și mai flexibil

```
package exemple.maiflexibil;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
public class Main {
	public static void main(String[] args) throws IOException {
		BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
		String aPluginClassName;
		while((aPluginClassName = bf.readLine()) != null) {
			try {
				Class loadedClass = Class.forName(aPluginClassName);
				Object anObject = loadedClass.newInstance();
				String methodName = bf.readLine();
				Method theMethod= loadedClass.getMethod(methodName,new Class[] {}); //Cautam o metoda cu acest nume si fara argumente
				theMethod.invoke(anObject, new Object[] {});  //Se pot da si argumente daca ar exista :)
			} catch(ClassNotFoundException e) {
				System.out.println("Plugin class not found!");
			} catch(InstantiationException | IllegalAccessException e) {
				System.out.println("Plugin class cannot be instantiated - concrete/accesible/no-arg constructors?");
			} catch (NoSuchMethodException e) {
				System.out.println("Method not found!");
			} catch (SecurityException e) {
				System.out.println("Security exception:" + e);
			} catch (IllegalArgumentException e) {
				System.out.println("Illegal arguments exception:" + e);
			} catch (InvocationTargetException e) {
				System.out.println("Invocation target exception:" + e);
			}
		}
	}
}
```

Exemplul 3 - Plugin și mai flexibil

```
package exemple.maiflexibil;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
public class Main {
	public static void main(String[] args) throws IOException {
		BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
		String aPluginClassName;
		while((aPluginClassName = bf.readLine()) != null) {
			try {
				Class loadedClass = Class.forName(aPluginClassName);
				Object anObject = loadedClass.newInstance();
				String methodName = bf.readLine();
				Method theMethod= loadedClass.getMethod(methodName,new Class[] {}); //Cautam o metoda cu acest nume si fara argumente
				theMethod.invoke(anObject, new Object[] {});  //Se pot da si argumente daca ar exista :)
			} catch(ClassNotFoundException e) {
				System.out.println("Plugin class not found!");
			} catch(InstantiationException | IllegalAccessException e) {
				System.out.println("Plugin class cannot be instantiated - concrete/accesible/no-arg constructors?");
			} catch (NoSuchMethodException e) {
				System.out.println("Method not found!");
			} catch (SecurityException e) {
				System.out.printl
			} catch (IllegalArgumen
				System.out.printl
			} catch (InvocationTar
				System.out.printl
			}
		}
	}
}
```

```
javac -d bin Main.java
```

```
java -cp bin:/Users/Pepi/Desktop/drop/ exemple.maiflexibil.Main
test
Plugin class not found!
```

Exemplul 3 - Plugin și mai flexibil (II)

```
package personal;
public class SomeClass {
	public void exec() {
		System.out.println("exec method is executed now :)");
	}
}
```

```
javac -d drop SomeClass.java
```

```
java -cp bin:/Users/Pepi/Desktop/drop/ exemple.maiflexibil.Main
test
Plugin class not found!
personal.SomeClass
metoda
Method not found!
personal.SomeClass
exec
exec method is executed now :)
```