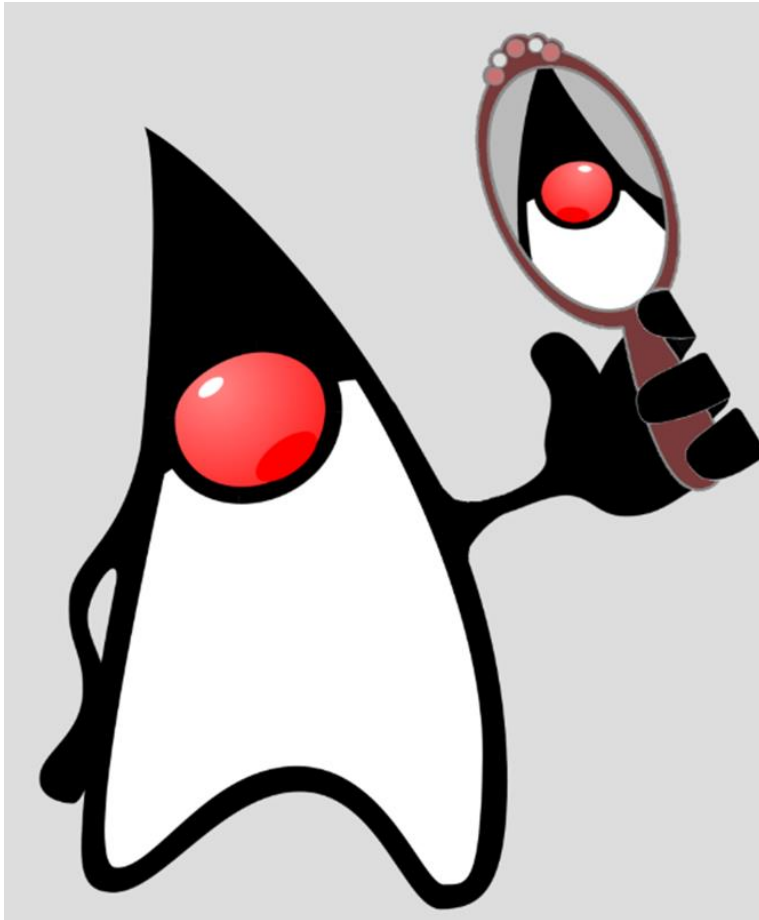


Metode avansate de programare

Informatică Româna, 2021-2022, Curs 11

- Reflexia (Reflection)
- Discutie cerinte lab si proiect extins
- Discutie detalii examinare

Reflection



Introspecția =
Capacitatea unui
program de a-și observa,
la execuție, propria
structură;

Încărcarea claselor în memorie

- Un program Java compilat este descris de o mulțime de fișiere cu extensia *.class* corespunzătoare fiecărei clase a programului.
- Aceste clase nu sunt încărcate toate în memorie la pornirea aplicației, ci sunt încărcate pe parcursul execuției acestuia, *atunci când este nevoie de ele*, momentul efectiv în care se realizează acest lucru depinzând de implementarea mașinii virtuale (JVM).
- Încărcarea claselor unei aplicații Java în memorie este realizată prin intermediul unor obiecte, denumite generic *class loader*.

Încarcarea **dinamică** a claselor în memorie

- Se refera la faptul ca nu cunoastem tipul acesteia decat la executia programului, moment in care putem solicita încarcarea sa, specificand numele său complet prin intermediul unui șir de caractere.
- Exista mai multe modalitati:
 - *loadClass* apelata pentru un obiect de tip `ClassLoader`
 - *Class.forName*

Observație

- În programarea orientată pe obiecte noțiunea de clasă și obiect sunt evident diferite
- În mecanismul de introspectie 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.

Încarcarea unei clase în memorie – metaclasa Class

Class.forName

Exemplu: clasa Task este reprezentată la execuția unui program de un obiect, instanță a clasei *Class*

```
Class aCls =Class.forName("domain.Task");  
System.out.println(aCls.getName());
```

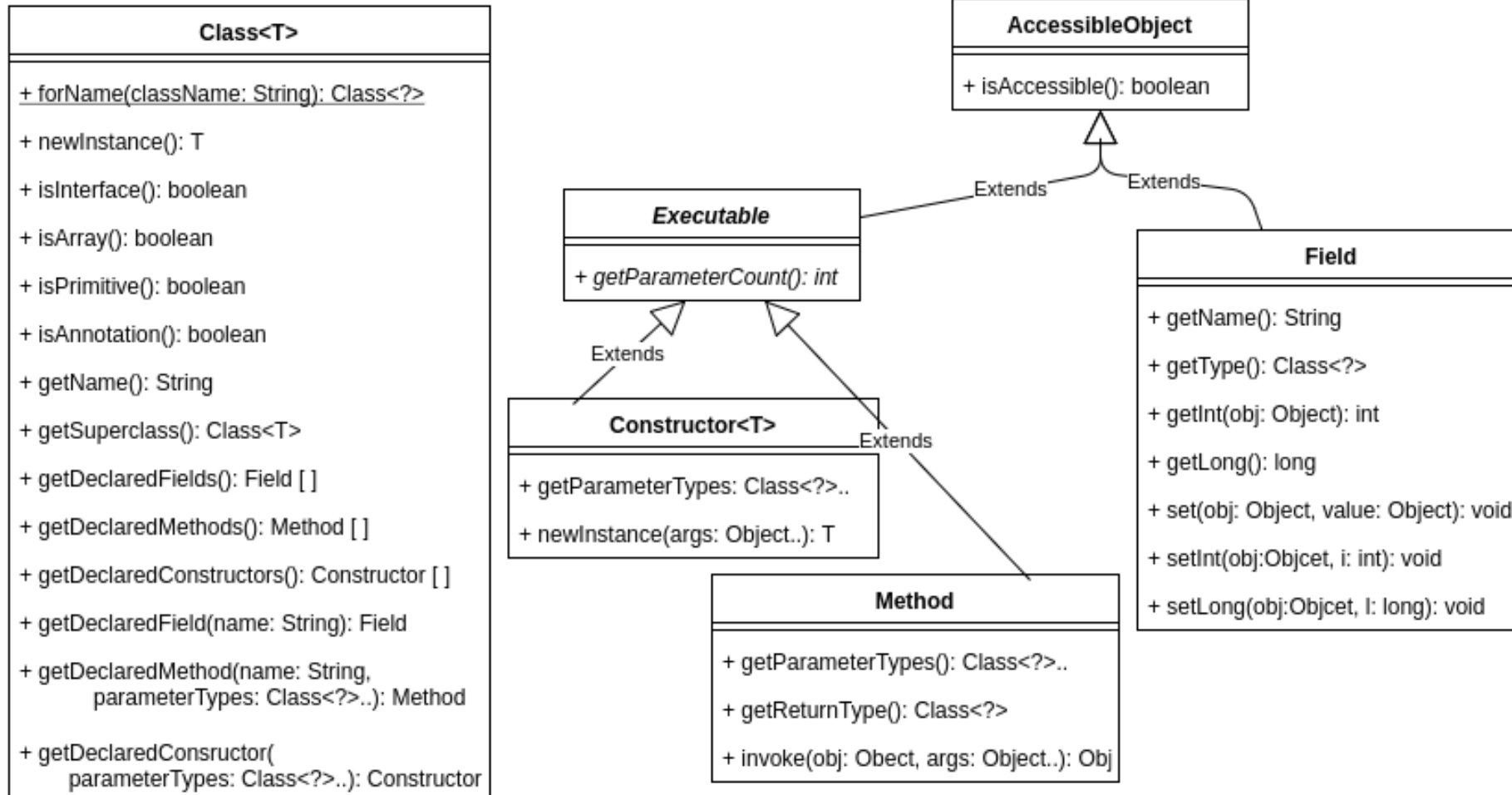
Task (from domain)
-taskID: int -description: String
«constructor»+Task(taskID: int, description: String) +getId(): Integer +setId(id: Integer): void +setDescription(description: String): void +getDescription(): String +execute(): void +hashCode(): int +equals(obj: Object): boolean +toString(): String +getClass(): Class



Class
...
+getName() : String +getFields() : Field[*] +getMethod() : Method[*] +getConstructors() : Constructor[*] <u>+forName(className : String) : Class</u>
...

- un obiect instanța a clasei *Class* reprezintă o clasă din cadrul programului

java.lang.reflect



Exemplu Reflectarea unei clase

```
public static void reflectClass(Class aClass) {
    String lines = "";
    lines += String.format("Class " + aClass.getName() + " having the following members:\n");

    for (Field aField : aClass.getDeclaredFields()) {
        lines += String.format("\tField name - %s :%s\n", aField.getName(), aField.getType());
    }
    for (Method aMethod : aClass.getDeclaredMethods()) {
        lines += String.format("\tMethod name - %s (): %s\n", aMethod.getName(), aMethod.getReturnType().getName());
        Parameter[] param = aMethod.getParameters();
        for (int i = 0; i < param.length; i++) {
            lines += String.format("\t\tParam %d - %s:%s\n", i + 1, param[i].getName(), param[i].getType());
        }
    }
    for (Constructor aConstructor : aClass.getConstructors()) {
        lines += String.format("\tConstructor name - %s:", aConstructor.getName());
        for (int i = 0; i < aConstructor.getParameters().length; i++) {
            Parameter param = aConstructor.getParameters()[i];
            lines += String.format("\t\tParam - %d: %s :%s\n", i + 1, param.getName(), param.getType());
        }
    }
    System.out.println(lines);
}
```


Task Mirror ☺

Class domain.Task having the following members:

```
Field name - taskID :int
Field name - description :class java.lang.String
Method name - equals (): boolean
    Param 1 - arg0:class java.lang.Object
Method name - toString (): java.lang.String
Method name - hashCode (): int
Method name - execute (): void
Method name - getId (): java.lang.Object
Method name - getId (): java.lang.Integer
Method name - setId (): void
    Param 1 - arg0:class java.lang.Integer
Method name - setId (): void
    Param 1 - arg0:class java.lang.Object
Method name - getDescription (): java.lang.String
Method name - setDescription (): void
    Param 1 - arg0:class java.lang.String
Method name - wait (): void
Method name - wait (): void
    Param 1 - arg0:long
    Param 2 - arg1:int
Method name - wait (): void
    Param 1 - arg0:long
Method name - getClass (): java.lang.Class
Method name - notify (): void
Method name - notifyAll (): void
Constructor name - domain.Task:      Param - 1: arg0 :int
    Param - 2: arg1 :class java.lang.String
```

Exercitii curs

- Instantierea unui obiect de un anumit tip folosind Reflection