



PROGRAMMAZIONE 2

5. Dynamic dispatch

PR2 2018-19

1



Cosa significa?

- La **dichiarazione** di una variabile non determina in maniera univoca il tipo dell'oggetto che la variabile riferisce
- Cerchiamo di capire quale è il problema

```
Class B extends class A  
// L'estensione riscrive il metodo m( )
```

- Supponiamo di creare un oggetto di tipo A e di tipo B

```
A a = new A( ) ... B b = new B()
```

e di fare diverse operazioni su a che coinvolgono anche aliasing con oggetti di tipo B (**a = b**). Poi invochiamo **a.m()**. Quale metodo è effettivamente invocato?

PR2 2018-19

2

Esempio (in un unico file)



```
public class DynamicBindingTest {
    public static void main(String[] args) {
        Vehicle vehicle = new Car(); // Il tipo statico è Vehicle
                                   // il tipo dinamico è Car
        vehicle.start(); //Quale metodo viene invocato?
                       // Quello di Car o quello di Vehicle?
    }
}
class Vehicle {
    public void start() {
        System.out.println("Inside start method of Vehicle");
    }
}
class Car extends Vehicle {
    public void start() {
        System.out.println("Inside start method of Car");
    }
}
```

3

Esempio (in un unico file)



```
public class DynamicBindingTest {
    public static void main(String[] args) {
        Vehicle vehicle = new Car(); // Il tipo statico è Vehicle
                                   // il tipo dinamico è Car
        vehicle.start(); //Quale metodo viene invocato?
                       // Quello di Car o quello di Vehicle?
    }
}
class Vehicle {
    public void start() {
        System.out.println("Inside start method of Vehicle");
    }
}
class Car extends Vehicle {
    public void start() {
        System.out.println("Inside start method of Car");
    }
}
```

Output:
Inside start method of Car

4

Intuizione



- Invocazione **obj.m()**
- A tempo di esecuzione viene utilizzato il tipo dinamico dell'oggetto **obj** per determinare nella gerarchia delle classi quale è il metodo più specifico da invocare

PR2 2018-19

5

-
- B è un *sottotipo* di A
 - A e B includono la definizione del metodo m

```
A a = new A ( );
B b = new B ( );
```

```
a.m( );
b.m( );
a = b;
a.m( );
```

Viene invocato il metodo della classe A

Viene invocato il metodo della classe B

Viene invocato il metodo della classe B



PR2 2018-19

6

Dynamic Dispatch

• Viene ricercato il metodo lungo la gerarchia a partire dal tipo **dinamico** dell'oggetto

```

A a = new A ( );
B b = new B ( );

a.m ( );
b.m ( );

```

PR2 2018-19 7

Dynamic Dispatch

• Viene ricercato il metodo lungo la gerarchia a partire dal tipo **dinamico** dell'oggetto

```

A a = new A ( );
B b = new B ( );

a.m ( );
b.m ( );
a = b;

```

Tipo statico di a è A
Tipo dinamico di a è B

PR2 2018-19 8

Static vs dynamic




- Il **compilatore** usa i tipi statici per determinare la correttezza delle invocazioni dei metodi
- La **macchina virtuale** usa il tipo dinamico per determinare l'effettivo metodo da invocare

La tabella dei metodi (versione intuitiva)



- Per comprendere gli esempi precedenti va estesa la ASM di Java con un'ulteriore componente: la tabella dei metodi
- La tabella contiene il codice dei metodi definiti nella classe, e tutte le componenti statiche definite nella classe stessa
- La tabella contiene un puntatore alla classe padre
- L'insieme delle tabelle è pertanto un albero (per quale ragione?)

Esempio



Class Table

Object
String toString(){...}
boolean equals...
...

Counter
extends Object
Counter() { x = 0; }
void incBy(int d){...}
int get() {return x;}

Decr
extends Counter
Decr(int initY) { ... }
void dec(){incBy(-y);}

```


public class Counter {
    private int x;
    public Counter() { x = 0; }
    public void incBy(int d) { x = x + d; }
    public int get() { return x; }
}

public class Decr extends Counter {
    private int y;
    public Decr (int initY) { super( ); y = initY; }
    public void dec() { incBy(-y); }
}

```

PR2 2018-19
11

Tabella dei metodi



- Le tabelle dei metodi sono allocate sullo heap (memoria dinamica)
- L'invocazione del metodo costruttore determina l'allocazione sullo heap della tabella dei metodi associata alla classe dell'oggetto creato (se non è già presente)
- Ogni oggetto sullo heap contiene un puntore alla tabella dei metodi del suo tipo **dinamico**

PR2 2018-19
12

Dispatch



- L'invocazione del metodo

`o.m()`

utilizza il puntatore alla tabella dei metodi per effettuare l'operazione di dispatch

- ricerca sulla gerarchia dell'oggetto a partire dalla tabella dei metodi associata al tipo dinamico dell'oggetto `o`
- da notare l'utilizzo di `this` per determinare l'oggetto che invoca il metodo

PR2 2018-19

13

```
public class Counter extends Object {
    private int x;
    public Counter( ) {
        super( );
        this.x = 0;
    }
    public void incBy(int d) { this.x = this.x + d; }
    public int get( ) { return this.x; }
}
public class Decr extends Counter {
    private int y;
    public Decr (int initY) {
        super( );
        this.y = initY;
    }
    public void dec() { this.incBy(-this.y); }
}
```

```
// nel main
Decr d = new Decr(2);
d.dec( );
int x = d.get();
```

14

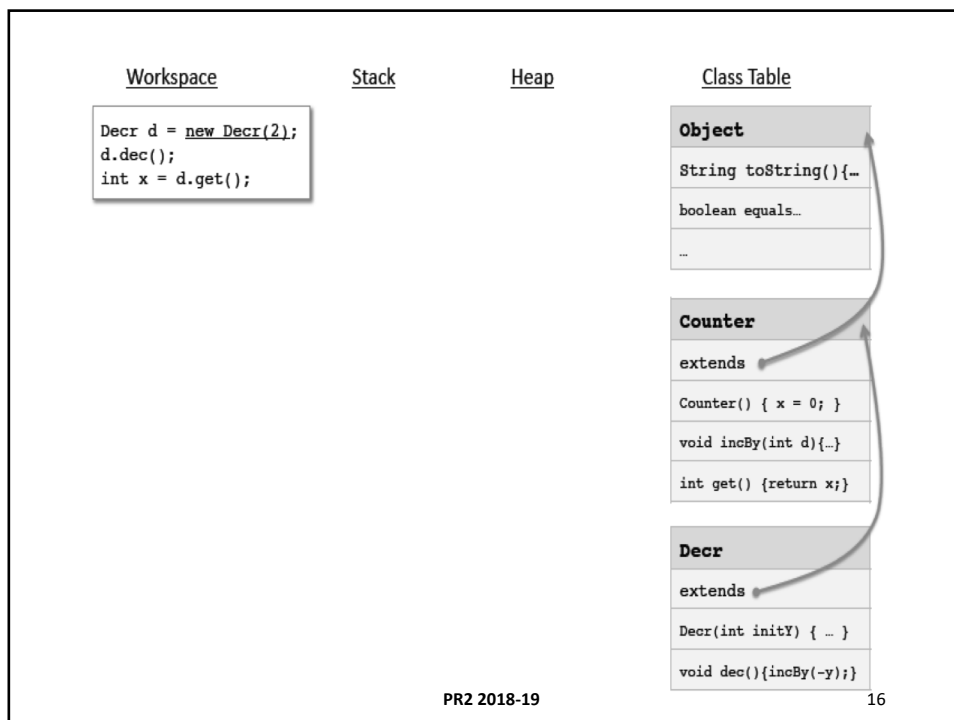




Animazione dell'esecuzione

PR2 2018-19

15



PR2 2018-19

16

