Realizzare Classi

Percorso formativo

- Programmare in Java:
  - Definire classi
  - Istanziare oggetti
- Imparare ad usare oggetti e classi predefiniti
- Imparare a definire nuove classi

Astrazione in Software Design

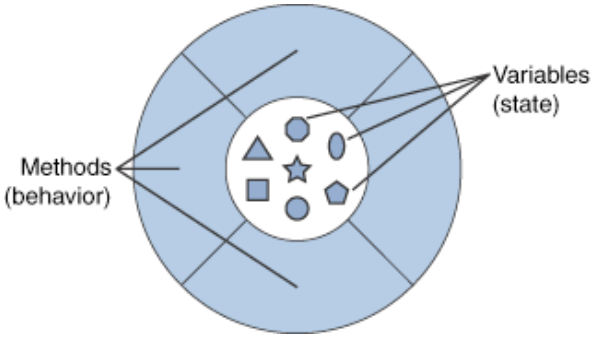
- In precedenza: i programmi manipolavano tipi di dati primitivi come numeri e caratteri
- Per il programmatore risulta essere molto complicato manipolare molte quantità primitive (si possono facilmente introdurre errori)
- Soluzione: Incapsulare le computazioni di routine in scatole nere (software)
- L'astrazione è usata per inventare tipi di dati di più alto livello (utilizzato nel processo di identificazione)
  - Nella programmazione OO gli oggetti sono scatole nere
- Incapsulamento: i programmatori usano un oggetto conoscendo il suo comportamento, ma non la sua struttura interna (utilizzato nell'implementazione)

Programmazione orientata agli oggetti

- Gi oggetti forniscono la base per le astrazioni
  - Gli elementi che descrivono la soluzione ad un problema sono implementati da oggetti equivalenti
- *Incapsulamento*: un programmatore che usa un oggetto conosce il suo comportamento ma non necessita conoscere la sua struttura interna
- Definire buone astrazioni non è semplice
  - È sicuramente possibile progettare cattivi programmi OO
- In una corretta progettazione (ad oggetti) prima si definiscono le classi e poi si implementano

Classi

- Il comportamento di un oggetto è descritto da una *classe*



- Ogni classe ha
  - Un'interfaccia pubblica
    - Insieme di metodi (funzioni) che si possono invocare per manipolare l'oggetto
    - Es.: `Rectangle(x_init,y_init,width_init,height_init)` metodo dell'interfaccia che crea un rettangolo (costruttore)
  - Un'implementazione nascosta
    - codice e variabili usati per implementare i metodi dell'interfaccia e non accessibili all'esterno della classe
    - Es.: `x, y, width, height`

La definizione di una classe

```
public class NomeDellaClasse {
        definizione di metodo
        definizione di metodo
        ...
}
```

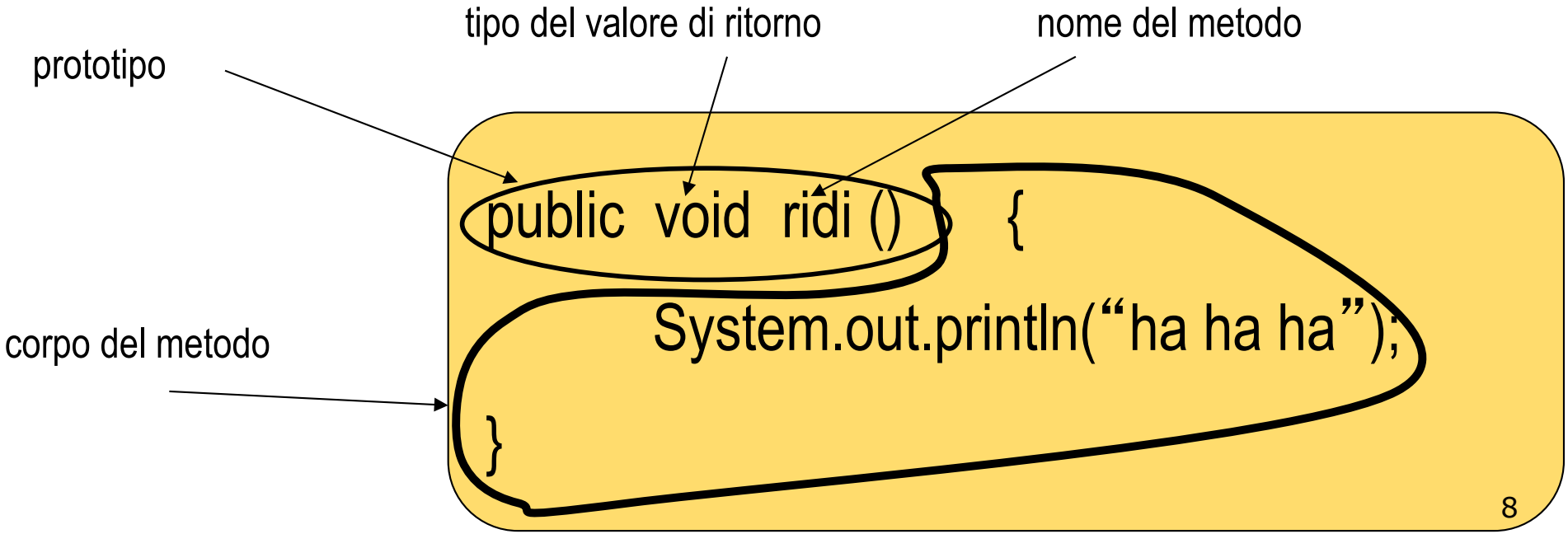
- Una classe contiene i suoi metodi (descrivono il comportamento comune alle istanze della classe)
- Le parentesi graffe aperte e chiuse fungono da delimitatori del contenuto di una classe

Un esempio

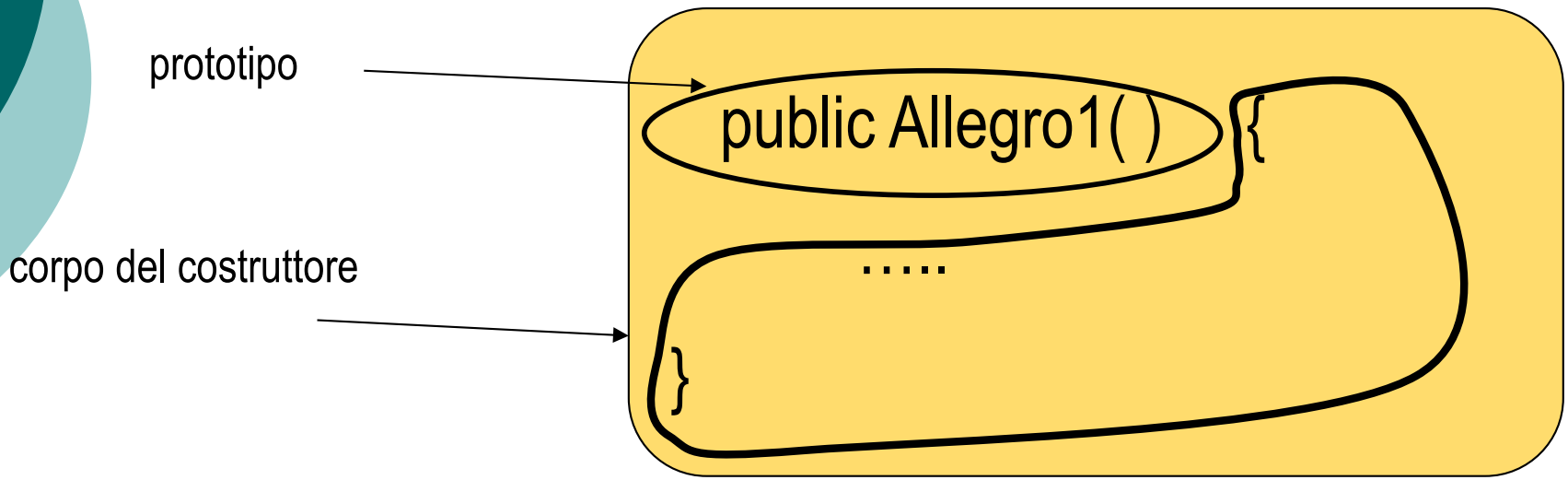
```
public class Allegro1 {
        public Allegro1() {
        }
        public void ridi() {
                System.out.println("haha");
        }
}
```

La definizione di un metodo

- Prototipo e corpo (**body**)
- Le parentesi graffe aperte e chiuse fungono da delimitatori del corpo del metodo



La definizione di un costruttore



- Nessun valore di restituzione
- Il nome coincide con quello della classe
- Invocato insieme a **new** per restituire un riferimento ad un oggetto appena creato

Utilizzare la classe **`Allegro1`**

1. Salvare la classe in un file (**`Allegro1.java`**)
2. Compilare la classe (**`javac Allegro1.java`**)
3. Scrivere un programma che utilizzi la classe **`Allegro1`**

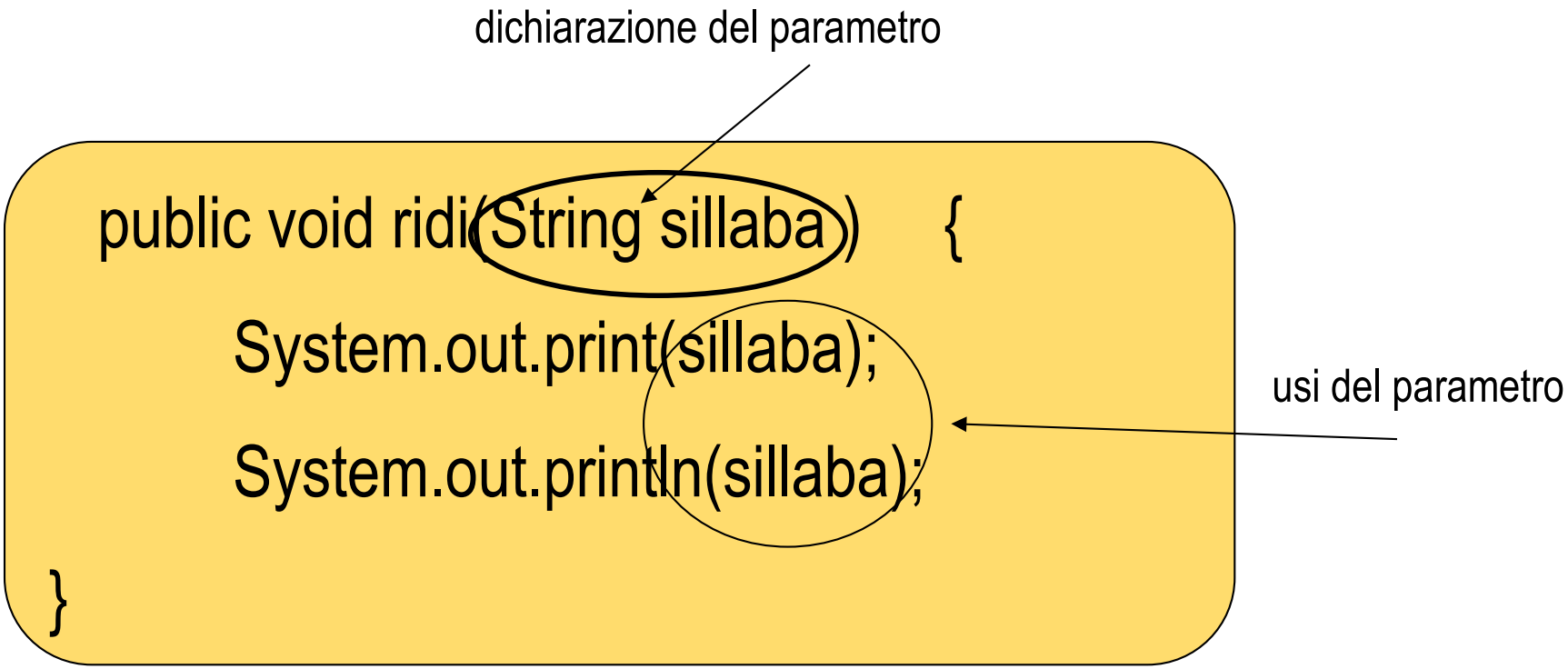
```
public class UsaAllegro1 {
        public static void main(String[] args) {
                Allegro1 x;
                x = new Allegro1();
                x.ridi();
                x.ridi();
        }
}
```

Metodi con argomenti

- Supponiamo che il mittente del messaggio **`ridi`** debba poter specificare la sillaba della risata
  - ha, ho, hee …

```
Allegro2 x;
x = new Allegro2();
x.ridi("ho");
x.ridi("hee");
```

Metodi con argomenti



- Il numero e tipo degli argomenti nel messaggio devono coincidere con quelli nel prototipo

Esempio con Overloading

```
class Allegro2 {
        public Allegro2() {
        }
        public void ridi() {
                System.out.println("haha");
        }
        public void ridi(String syl) {
                System.out.print(syl);
                System.out.println(syl);
        }
}
```

- Notare che il metodo **`ridi`** è overloaded

Oggetti con memoria

- Supponiamo di voler fornire al metodo costruttore un argomento String che rimpiazzi “ha” come sillaba di default per la risata

- ```
  Allegro3 x;
  x = new Allegro3(“ho”);
  x.ridi();                  // hoho
  x.ridi("hee");             // heehee
  x.ridi();                  // hoho
  ```
  ```

Oggetti con memoria

- Costruttore: `public Allegro3(String syl) …`
- Problemi:
  - Gli argomenti di un metodo esistono solo durante l'esecuzione del metodo
  - Gli argomenti di un metodo sono visibili solo al metodo
- Come fare in modo che il metodo `ridi` possa conoscere il valore della `syl` ?
- Fornire agli oggetti la capacità di memorizzazione

  Variabili di istanza (instance variables)

Variabili di istanza

- Una variabile dichiarata all'interno di una classe ma al di fuori di qualsiasi metodo
  - Accessibile a tutti i metodi dell'oggetto
  - Lo scopo è di memorizzare le informazioni necessarie ai metodi che devono essere preservate tra diverse invocazioni
  - Ciascun oggetto ha le proprie variabili di istanza le quali hanno i propri valori
  - Stato di un oggetto: i valori delle variabili di istanza
  - Tipicamente sono inizializzate dal costruttore

Esempio

```
public class Allegro3 {
        public Allegro3(String syl) {
                defaultSyl = syl;
        }
        public void ridi() {
                System.out.print (defaultSyl);
                System.out.println (defaultSyl);
        }
        public void ridi(String syl) {
                System.out.print(syl);
                System.out.println(syl);
        }
        private String defaultSyl;
}
```

Le variabili di istanza possono essere usate da tutti i metodi

Dichiarazione di una variabile di istanza

…miglioriamo ancora …

```
public class Allegro4 {
        public Allegro4() {
                defaultSyl = "ha";
        }
        public Allegro4(String syl) {
                defaultSyl = syl;
        }
        public void ridi() {
                System.out.print (defaultSyl);
                System.out.println(defaultSyl);
        }
        public void ridi(String syl) {
                System.out.print(syl);
                System.out.println(syl);
        }
        public void ridi(String s1, String s2) {
                System.out.print(s1);
                System.out.println(s2);
        }
        private String defaultSyl;
}
```

… utilizzatore …

```
public class RidiamoUnPoco {
        public static void main(String[] a) {
                System.out.println(“Vivi allegramente!");
                Allegro4 x,y,z;
                x = new Allegro4("yuk");
                y = new Allegro4("harr");
                z = new Allegro4();
                x.ridi();
                x.ridi("hee");
                y.ridi();
                z.ridi();
        }
}
```