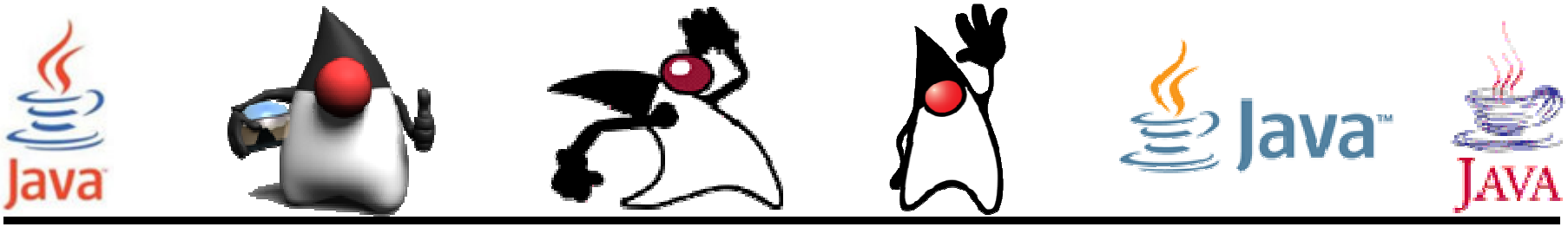


Polimorfismo

Pedro Corcuera

Dpto. Matemática Aplicada y
Ciencias de la Computación

Universidad de Cantabria

corcuerp@unican.es

Objetivos

- Comprender el mecanismo del polimorfismo
- Aprender a utilizar el polimorfismo en clases

Índice

- ¿Qué es polimorfismo?
- Beneficios del Polimorfismo
- Ejemplos

¿Qué es polimorfismo?

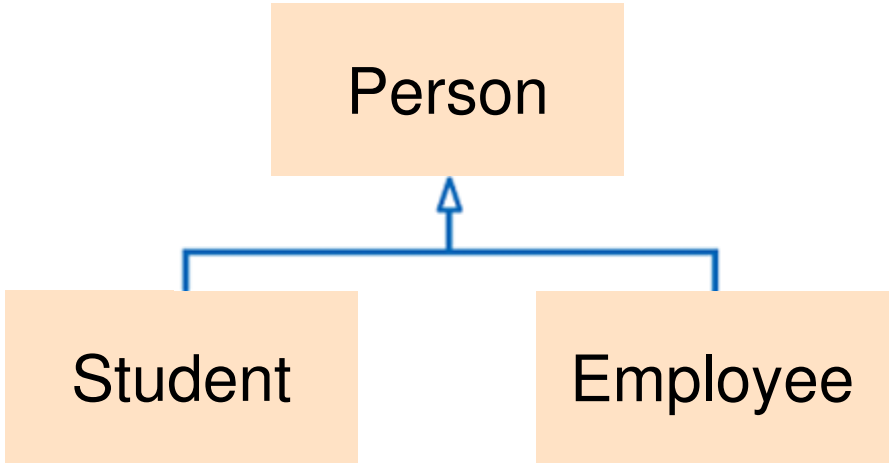
- Polimorfismo se refiere a la habilidad para aparecer en varias formas
- Polimorfismo en programas Java significa:
  - La habilidad de una variable referencia para cambiar su comportamiento de acuerdo a la instancia del objeto que contiene
  - Esto permite que múltiples objetos de diferentes subclases sea tratados como objetos de una superclase única, mientras que automáticamente se selecciona los métodos apropiados a aplicar a un objeto en particular basado en la subclase a la que pertenece

Ejemplo de polimorfismo

- Por ejemplo, dada una clase base *shape*, el polimorfismo permite al programador definir diferentes métodos *area* para cualquier número de clases derivadas, tales, como *circles*, *rectangles*, y *triangles*
  - El método area de circle, revtangle y triangle se implementa de manera diferente
- No importa qué forma tiene un objeto, aplicándole el método *area* devolverá el resultado correcto

Ejemplo 1: polimorfismo

- Dada la clase padre `Person` y la clase hija `Student`, se añade otra subclase `Person` que es `Employee`
- La gráfica de la jerarquía de clases es:



Ejemplo 1: polimorfismo

- Se puede crear una referencia que es del tipo de la superclase, *Person*, hacia un objeto de su subclase *Student*

```
public static main( String[] args ) {
    Student studentObject = new Student();
    Employee employeeObject = new Employee();

    Person ref = studentObject; //referencia Person
                                // al objeto Student
  // Llamada de getName() de la instancia de objeto Student
  String name = ref.getName();
}
```

Ejemplo 1: polimorfismo

- Supongamos ahora que hay un método getName en la superclase Person y que este método es sobrescrito en las subclases Student y Employee

```
public class Student {  public String getName() {
    System.out.println("Nombre estudiante:" + name);
    return name;
  }
}

public class Employee {  public String getName() {
    System.out.println("Nombre empleado:" + name);
    return name;
  }
}
```

Ejemplo 1: polimorfismo

- Volviendo al método main, cuando se invoca el método getName de la referencia Person ref, el método getName del objeto Student será llamado
- Si se asigna ref a un objeto Employee, el método getName de Employee será llamado

Ejemplo 1: polimorfismo

```
public static main (String[] args) {
  Student studentObject = new Student();
  Employee employeeObject = new Employee();

  //ref apunta a un objeto Student
  Person ref = studentObject;

  //metodo getName() de la clase Student es invocado
  String temp= ref.getName();
  System.out.println(temp);

  //ref apunta ahora a un objeto Employee
  ref = employeeObject;

  //metodo getName() de la clase Employee es invocado
  String temp = ref.getName();
  System.out.println(temp);
}
```

Ejemplo 2: polimorfismo

- Otro ejemplo que ilustra el polimorfismo es cuando se tarta de pasar una referencia a métodos como parámetros
- Supongamos que tenemos un método estático printInformation que recibe una referencia Person como parámetro

```
public static printInformation ( Person p ){
  // Se invocara el metodo getName() de la
  // instancia actual del objeto que es pasado
  p.getName();
}
```

Ejemplo 2: polimorfismo

- Si se pasa una referencia de tipo Employee y tipo Student al método printInformation en tanto que es una subclase de la clase Person

```
public static main( String[] args ){
  Student studentObject = new Student();
  Employee employeeObject = new Employee();

  printInformation( studentObject );
  printInformation( employeeObject );
}
```

Beneficios del Polimorfismo

- Simplicidad
  - Si se necesita escribir código que trata con una familia de subtipos, el código puede ignorar los detalles específicos de tipo y sólo interactuar con el tipo base de la familia
  - Aun cuando el código piense que está usando un objeto de la clase base, la clase del objeto podría ser la clase base o cualquiera de sus subclases
- Extensibilidad
  - Se pueden añadir subclases posteriormente a la familia de tipos, y los objetos de estas nuevas subclases podría trabajar con el código existente

Tres formas de polimorfismo en un programa Java

- Overriding de método
  - Métodos de una subclase sobreescriben los métodos de una superclase
- Overriding de método (implementación) de los métodos abstractos
  - Métodos de una superclase implementa los métodos abstractos de una clase abstracta
- Overriding de método (implementación) a través de interface
  - Métodos de una superclase implementa los métodos abstractos de la interface