Programación 2

Lenguaje Java - Conceptos claves (*3era parte*)

Típos de Clases

Profesor: Eduardo Godoy.

eduardo.gl@gmail.com

Escuela de Ingeniería Civil Informática.
Universidad de Valparaíso.

31 de agosto de 2017

Contenido

Clase abstracta

Consideraciones

- Éste tipo de clases solo existe para que otra extienda de ella.
- Se define como clases abstracta a una clase que represente al "padre"dentro de un esquema de herencia o a también a una clase abstracta que permite generalizar ciertos comportamientos que son identicos dentro de un grupo de clases.
- Una clase abstracta nunca puede ser instanciada.

Clase abstracta

Método Abstracto.

Método declarado en una clase la cual no lo implementa. Al definirse un método como abstracto dentro de una clase, implica que la clase que lo contiene se también transforma en abstracta.

Ejemplo.

Clase Abstracta.

```
import java.util.*;

public abstract class Instrumento {
    public abstract void tocar();
    public String tipo() {
  return "Instrumento";
    }
    public abstract void afinar();
}
```

Ejemplo.

Implementaciíon de clase abstracta.

```
import java.util.*;

public class Guitarra extends Instrumento {
    public void tocar() {
  System.out.println("Guitarra.tocar()");
    }
    public String tipo() { return "Guitarra"; }
    public void afinar() {}
}
```

Ejemplo.

Implementaciíon de clase abstracta.

```
import java.util.*;

public class Piano extends Instrumento {
    public void tocar() {
  System.out.println("Piano.tocar()");
    }
    public String tipo() { return "Piano"; }
    public void afinar() {}
}
```

Ejemplo.

Implementaciíon de clase abstracta.

```
import java.util.*;

public class Saxofon extends Instrumento {
    public void tocar() {
  System.out.println("Saxofon.tocar()");
    }
    public String tipo() { return "Saxofon"; }
    public void afinar() {}
}
```

Ejemplo.

Clase Main

```
import java.util.*;

public public class MainMusica {

    // No importa el tipo de Instrumento,
    // seguira funcionando debido a Polimorfismo:
  static void afinar(Instrumento i) {
     i.tocar();
  }

  static void afinarTodo(Instrumento[] e) {
     for(int i = 0; i < e.length; i++)
       afinar(e[i]);
     }
  }
```

Ejemplo.

Clase Main

```
  public static void main(String[] args) {
    // Declarar un Arreglo SIN INSTANCIAS es valido en
        Clases Abstractas
    Instrumento[] orquesta = new Instrumento[5];
    // Generar una INSTANCIA de una la Clase Abstracta
        no es valido
   // Instrumento nuevo = new Instrumento();
    int i = 0;
    orquesta[i++] = new Guitarra();
    orquesta[i++] = new Piano();
    orquesta[i++] = new Saxofon();
    orquesta[i++] = new Guzla();
    orquesta[i++] = new Ukelele();
    afinarTodo(orquesta);
  }
}
```

Interface

- Son superclases en las cuales la totalidad de sus métodos se definen como métodos abstractos, que deben ser implementados en las subclases en las cuales la interface es implementada.
- La interface define y conoce que métodos tiene, pero no sabe como es su implementacińn.
- Por ejemplo la Interfaces Animal define el método eat() sin conocer la lógica de como cada tipo de animal se alimenta, luego todas las clases de animal que implementen ésta interface definiran de distinta forma el método eat() dependiendo de la naturaleza de cada animal.
- En una interface en Java sus métodos siempre serán públicos y abstractos.

Ejemplo.

Clase Main

```
import java.util.*;

public interface IntegranteSeleccionFutbol {
  void concentrarse();
  void viajar();
  void entrenar();
  void jugarPartido();
}
```

Ejemplo.

Clase Main

```
public class Futbolista implements
    IntegranteSeleccionFutbol {
  private int dorsal;
  private String demarcacion;
  // Constructor, getter y setter
  @Override
  public void entrenar() {...}
  @Override
  public void jugarPartido(){...}
  @Override
  public void entrevista() {...}
}
```

Ejemplo.

Clase Main

```
public class Entrenador implements
    IntegranteSeleccionFutbol {
  private int identificacion;
  private String nombre;
  // Constructor, getter y setter
  @Override
  public void entrenar() {...}
  @Override
  public void jugarPartido(){...}
  @Override
  public void entrevista() {...}
}
```

Final class

Consideraciones

- La palabra reservada final que se antepone al nombre de una clase quiere decir que dicha clase no puede tener subclases, es decir, que no puede ser extendida .
- Se recomienda hacer que una clase final solo si se requiere de absoluta garantia de que sus métodos no será re utilizados, o sobrescrito.

Ejemplo.

Clase Main

```
public final class NoMePuedenExtender {
  //ERROR DE COMPILACION
}
```

Ejemplo.

Clase Main

```
import java.util.*;

public class OtraClase extends NoMePuedenExtender {

}
```