Clases Abstracta e Interfaces

Dr. Mario Marcelo Berón

Clase Abstracta

Concepción

1. Una clase abstracta es una clase que se declara como *abstract* dicha clase puede o no incluir métodos abstractos.
2. Una clase abstracta no se puede instanciar pero si se puede utilizar como superclase. La imposiblidad de crear instancias puede surgir por la naturaleza de la clase o bien por restricciones impuestas por el programador.

Clase Abstracta Declaración

La creación de una clase abstracta se realiza de la siguiente manera:

```
public abstract class NombreClase{
    .....
}
```

Método Abstracto

Concepción

Un método abstracto es aquel que se declara con *abstract* y no tiene implementación.Cuando se declara un método abstracto, la clase será abstracta y las subclases estarán forzadas a implementarlo caso contrario también serán abstractas.

Método Abstracto Declaración

Un método abstracto se declara de la siguiente manera:

```
modDeAcc abstract tipoDeRetorno nombre(param.){
    .....
}
```

Ejemplo

```
abstract void moverA(double deltaX, double deltaY);
```

Clases con Métodos Abstractos

Si una clase incluye métodos abstractos, entonces la clase se debe declarar como abstracta.

Ejemplo

```
public abstract class ObjetoGráfico {
    // declarar campos
    // declarar métodos no abstractos
    abstract void dibujar();
}
```

Notas y Comentarios

Una subclase de una clase abstracta provee implementaciones para todos los métodos abstractos de su clase padre. Sino lo hace la subclase se debe declarar como abstracta.

Clases Abstractas

El modificador *final* además de utilizarse para declarar constantes se puede utilizar para:

- Cortar la herencia.

```
modAcc final class nombreClase{
    ....
}
```

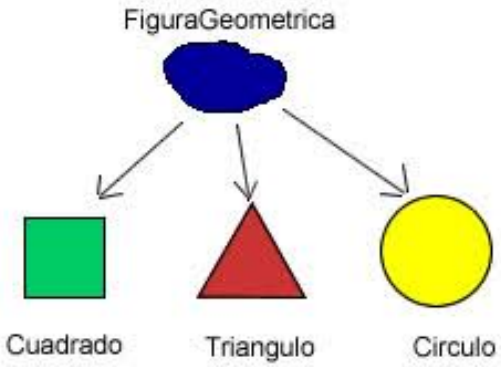
  no se pueden crear subclases.

- Evitar ocultar métodos

```
modAcc final tipo nombre(param.){
    ....
}
```

  no se puede sobre escribir el método en la subclase.

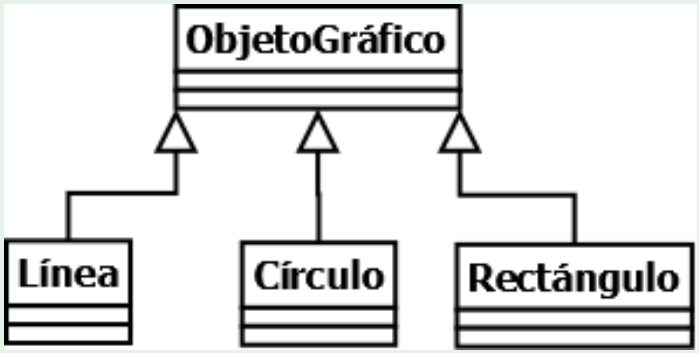
Clases Abstractas



- La clase *Object* es la raíz de la jerarquía de clases de Java.
- El método *toString* pertenece a la clase *Object* y por lo tanto se hereda a todas las clases y puede ser reescrito. Se utiliza con los System.out.println para imprimir información de los objetos.
- Emboltorios (wrapper) son utilizados para poder colocar en las colecciones tipos primitivos tales como int o char. Para cada tipo primitivo existe un emboltorio una clase que envuelve a un tipo básico.
- Autoboxing es una técnica que permite convertir un tipo primitivo a su envoltorio correspondiente.

Clases Abstractas

Ejemplo



Clases Abstractas

Ejemplo

```
abstract class ObjetoGráfico {
  private int x, y;
  ...
  void moverA(int deltaX, int deltaY) {
      ...
  }
  abstract void dibujar();
  abstract void cambiarTamaño();
}
```

Clases Abstractas

Ejemplo

```
class Círculo extends ObjetoGráfico {
    void dibujar() {
        ...
    }
    void cambiarTamaño() {
        ...
    }
}
class Rectángulo extends ObjetoGráfico {
    void dibujar() {
        ...
    }
    void cambiarTamaño() {
        ...
    }
}
```

Interface

Idea

Realidad

- Imagine una sociedad futurista donde los autos manejados por robots pueden transportar pasajeros.
- Las empresas de autos escriben software para: parar, encender, acelerar, doblar a la izquierda, etc.

Interface

Idea

Realidad

- Otros grupos industriales construyen sistemas de computadoras que capturan datos desde un GPS y transmisiones inalámbricas de las condiciones del tráfico.
- Las dos empresas deben publicar una interfaz estándar que declare qué métodos pueden ser invocados para que el auto se movilice.

Interface

Idea

Realidad

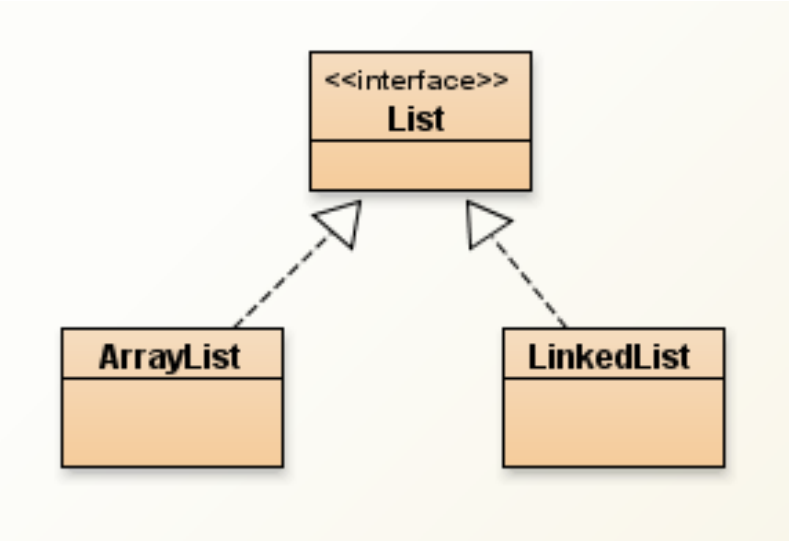
- La empresa que recolecta datos del GPS y de información de tránsito puede invocar los métodos descritos en la interfaz de comandos del auto.
- Ningún grupo industrial necesita conocer cómo se implementó el software del otro grupo.

Interface

Concepción

Una Interface es un tipo de referencia, similar a una clase, que puede contener solo constantes, declaraciones de variables, signaturas de métodos, métodos por defecto, métodos estáticos. Los únicos métodos que pueden ser implementados son los métodos estáticos y los métodos por defecto. Las interfaces no pueden ser instanciadas ellas solo pueden ser implementadas por clases o extendidas por otras interfaces.

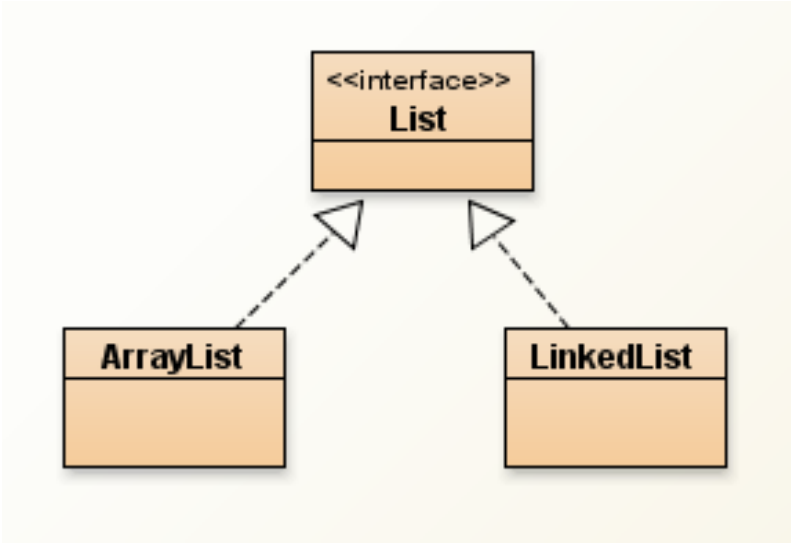
Interfaces



Interface:contiene métodos abstractos.

- Cuando se desea que una utilice los métodos de una interface se usa la palabra *implements* en lugar de *extends*.
- Cuando una clase implementa una interface tiene el mismo tipo que la interface tan bien como el tipo de su propia clase.

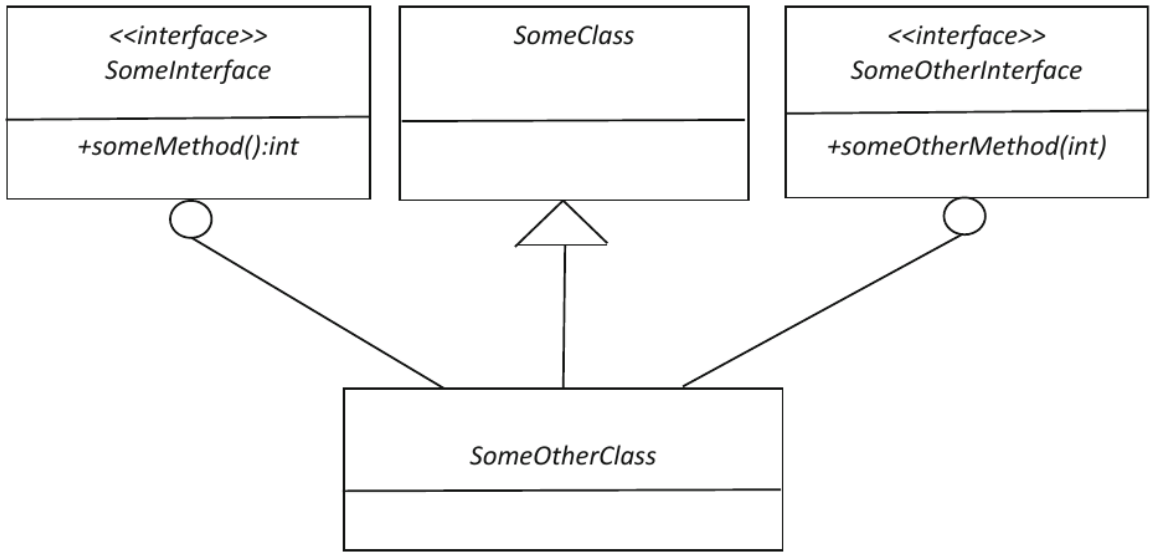
Interfaces



Importante

Implementar una interface es como hacer un contrato con el usuario de una clase. Dicho contrato garantiza que la clase tendrá métodos particulares.

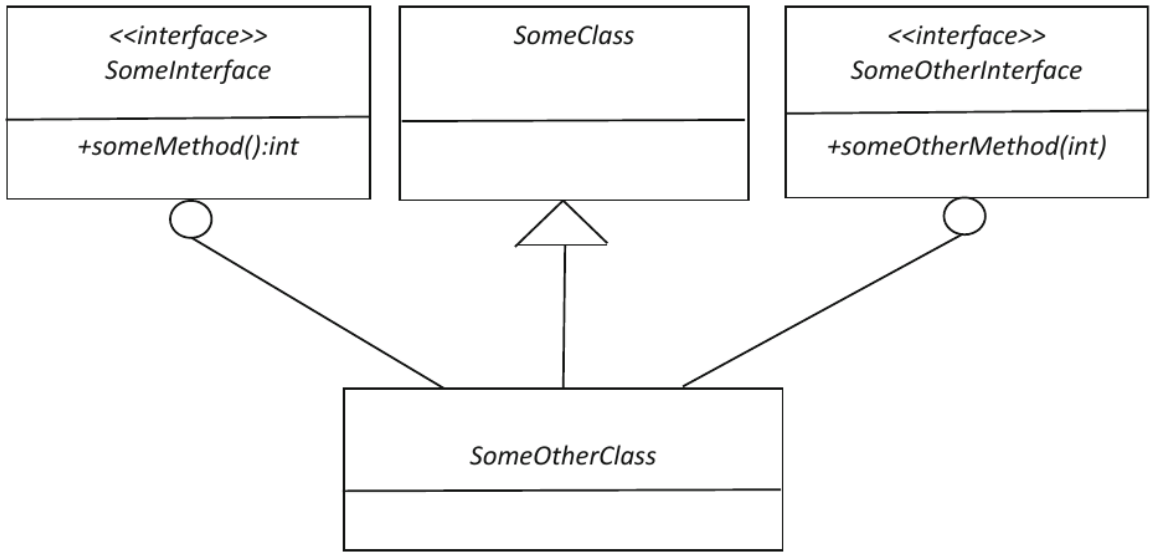
Interfaces



Comentarios Importantes

- En UML una interface se denota con la palabra interface encerrada entre dos corchetes angulares.
- El círculo del lado de la interface indica que la clase implementa la interface.
- Una clase puede implementar más de una interface pero solo puede heredar de una clase.

Interfaces



Comentarios Importantes

- Dado que los métodos que se encuentran en la interface son abstractos los mismos no son codificados en la interface pero si en la clase que las implementa. Razón por la cual no hay conflictos como en el caso del a herencia múltiple.
- Una interface puede contener *métodos static* y *métodos default*.

Interfaces

Definición de una Interface

```
modAcc interface nombre {
    ....
}
```

Ejemplo

```
public interface Checkable {
    public boolean check();
}
```

Comentarios

En una interface los métodos no es necesario indicar que los métodos que no son estáticos y por defecto son abstractos dado que por definición lo son.

Interface

Ejemplo

```
public interface OperarAuto {
  // Declaraciones de constantes
  // Signaturas de métodos
  // Una enumeración con valores izquierdo
  // y derecho
  int doblar(Dirección dirección, double radio,
  double velocidadDeComienzo,
  double velocidadDeFin);
  int cambiarCarriles(Dirección dirección,
  double velocidadDeComienzo,
  double endDeFin);
  ......
}
```

Interface

Uso

Para usar una interface se debe escribir una clase que implementa la interface.

Ejemplo

```
public class OperarBMW implements OperarAuto {
  /* las signaturas de OperarAuto con
  implementaciones, por ejemplo: */
  int doblar(Dirección dirección,
  double radio,double
  velocidadDeComienzo,
  double velocidadDeFin) {...}
  /* Otros miembros que se necesiten, como por
  ejemplo clases no visibles a los
  clientes de la interface */
}
```

Definición de una Interface - Extendido

La declaración de una interface consiste de modificadores de acceso, el nombre de la interface, una lista separadas por coma de las interfaces padres, y el cuerpo de la interface.

Ejemplo

```
public interface InterfaceAgrupada extends
Interface1,
Interface2, Interface3 {
  // declaraciones constantes
  // base de logartimo natural
  static final double E = 2.718282;
  // signatura de métodos
  void hacerAlgunaCosa (int i, double x);
  int hacerAlgoMás(String s);
}
```

Definición de una Interface

1. El identificador de acceso *public* indica que la interface puede ser usada por cualquier clase en cualquier paquete.
2. Una interface puede extender otras interfaces de la misma forma que una clase puede extender a otra clase.
3. Una interface puede extender cualquier número de interfaces. La declaración de la interface incluye una lista separadas por coma de todas las interfaces que extiende.

El Cuerpo de la Interface

1. El cuerpo de la interface contiene métodos abstractos, métodos por defecto y métodos estáticos. Un método dentro de la interface está seguido por un punto y coma y no contiene una implementación.
2. Los métodos por defecto están definidos con el modificador default.
3. Los métodos estáticos están definidos con el modificador static.
4. Todos los métodos abstractos, por defecto y estáticos en una interface son implícitamente públicos.
5. Una interface puede contener declaraciones constantes. Todas los valores constantes definidos en una interface son implícitamente públicos, estáticos y final.

Implemetación de una Interface

1. Para definir una clase que implementa a una interface se debe incluir la clausula *implements* en la declaración de la clase.
2. Una clase puede implementar más de una interface por lo tanto la clausula *implements* es seguida por una lista separada por comas de interfaces a las cuales implementa la clase.
3. Por convención la clausula *implements* sigue a la clausula *extends*.

Evolución de una Interface

Se puede realizar básicamente de las siguientes formas:

1. Por extensión de la interface.
2. Definiendo métodos por defecto.

Evolución de una Interface

Por Extensión de la Interface

```
public interface Hacer {
  void hacerAlgo(int i, double x);
  int hacerAlgoMás(String s);
}

public interface HacerExtendido extends Hacer {

  boolean hacerTrabajo(int i, double x,
  String s);

}
```

Interfaces

Método default: Es un método regular implementado en la clase y automáticamente heredado por todas las clases que implementan a la interface. La implementación de la clase puede anularlo si así lo necesita.

Métodos por Defecto

Incorporar un método por defecto a la interface significa que toda clase que implemente la versión previa de esa interface tiene que cambiar.

Evolución de una Interface

Definición de Métodos por Defecto

```
public interface Hacer {
  void hacerAlgo(int i, double x);
  int hacerAlgoMás(String s);
  default boolean hacerTrabajo(int i, double x,
  String s) {...}
}
```

Notas y Comentarios

Se debe proveer la implementación de los métodos por defecto y estáticos. Los usuarios que usan interfaces con métodos por defecto o estáticos no necesitan recompilar o acomodar los métodos que las usan.

Evolución de una Interface

Síntesis

1. Una interface puede contener signatura de métodos, métodos por defecto, métodos estáticos y definiciones de constantes, campos estáticos.
2. Una clase que implementa una interface debe implementar todos los métodos declarados en la interface.
3. Una interface puede ser usada en los mismos lugares en los cuales se usa un tipo.

Clases Abstractas vs. Interfaces

1. Las Clases Abstractas y las Interfaces son similares:
   - No son instanciables.
   - Pueden contener métodos con o sin implementación.
2. Con las clases abstractas se pueden declarar campos que no son estáticos y final, y definir métodos concretos públicos, privados y protegidos.
3. Se puede extender solo una clase abstracta pero se pueden implementar muchas interfaces.

Clases Abstractas vs. Interfaces

¿Cuándo se usa una clase abstracta o una interface?

1. Una clase abstracta se usa cuando:
   - Se desea compartir código entre diferentes clases estrechamente relacionadas.
   - Las clases que extiendan a la clase abstracta tengan muchos métodos y campos en común, o se requieran modificadores de acceso protegido y privado.
   - Se desea declarar campos no estáticos y no finales. Esto permite que se pueda acceder y modificar el estado del objeto al cual ellos pertenecen.
2. Una interface se usa cuando:
   1. Clases poco relacionadas implementen una interface.
   2. Se desea especificar el comportamiento de un tipo particular, pero no se está interesado en como se implementa ese comportamiento.