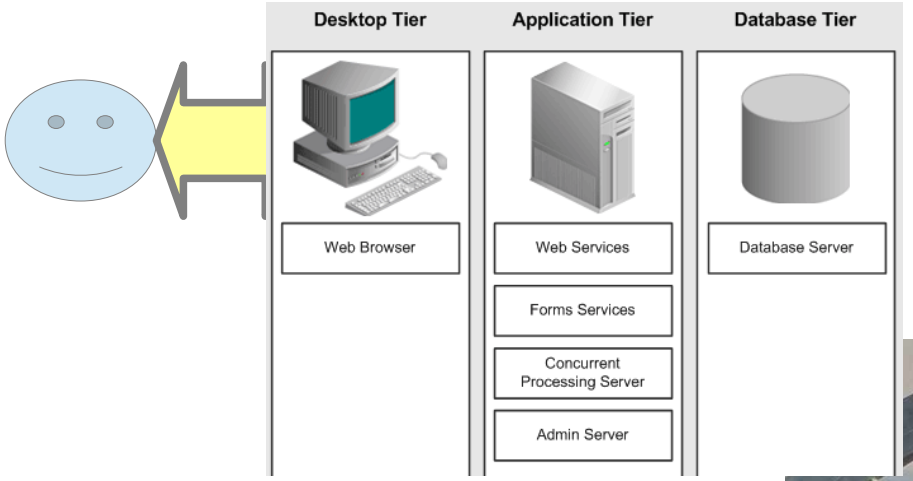
Intro POO (VIII)
- Interfaces

Recordando...

- Uso de *packages*
- Manejo de Excepciones
  - Bloque try-catch
- Uso de archivos

¿Qué son las interfaces?



Interfaces

- Es un **conjunto de declaraciones de métodos** *(sin definición)* que definen un tipo de conducta para las clases que las utilicen.
- Todas las **clases que implementan** una determinada **interface** están:
  - **obligadas** a proporcionar una **definición** de todos los métodos de la interface
  - adquieren una conducta o modo de funcionamiento.

Interfaces

- Una clase puede implementar una o varias interfaces.
- Para indicar que una clase implementa una o más interfaces se usa ***implements***.

```
class Circulo extends Figura implements Dibujable
{ ... }
```

Interfaces

- Una interfaz es la descripción de algún servicio que posteriormente alguna clase puede implementar.
- En algunos casos puede ser utilizado para implementar herencia múltiple en Java.
  - La cual no puede implementarse vía ***extends***

Interfaces

- Cada **interface public** debe ser definida en un archivo ***.java** con el **mismo nombre** de la **interface**.
- Obs: Los nombres de las interfaces suelen comenzar también con mayúscula.
- Las **interfaces sólo admiten** los modificadores de acceso **public y package**.
  - Si la interface no es public no será accesible desde fuera del package.
  - Los *métodos* declarados en una interface son siempre *public y abstract*, de modo implícito.

Interfaces

```
public interface Dibujable {
    public void setPosicion(double x, double y);
    public void dibujar(Graphics dw);
}
```

La interface anterior debería guardarse en el archivo:

- **Dibujable.java**

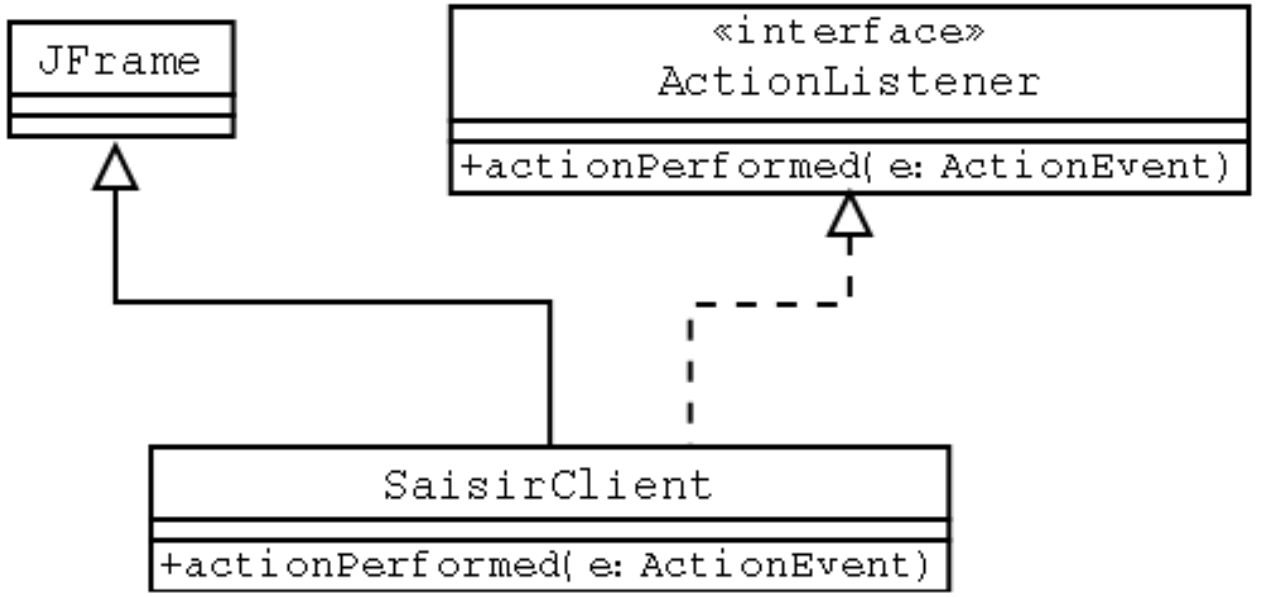
Interfaces

```
interface Bicycle {

    //  wheel revolutions per minute
    void changeCadence(int newValue);

    void changeGear(int newValue);

    void speedUp(int increment);

    void applyBrakes(int decrement);
}
```

```
class ACMEBicycle implements Bicycle {

    int cadence = 0;
    int speed = 0;
    int gear = 1;

   // The compiler will now require that methods
   // changeCadence, changeGear, speedUp, and applyBrakes
   // all be implemented. Compilation will fail if those
   // methods are missing from this class.

    void changeCadence(int newValue) {
         cadence = newValue;
    }

    void changeGear(int newValue) {
         gear = newValue;
    }

    void speedUp(int increment) {
         speed = speed + increment;
    }

    void applyBrakes(int decrement) {
         speed = speed - decrement;
    }

    void printStates() {
         System.out.println("cadence:" +
             cadence + " speed:" +
             speed + " gear:" + gear);
    }
}
```

Interfaces

- Modelando una situación usando ***interfaces*** con UML.



¿Qué “significa” este modelo?

Interfaces

- ¿Qué diferencia una **interface** de una **abstract class**?
  - Una clase no puede heredar de dos clases abstract, pero sí puede heredar de una clase abstract e **implementar una o más interfaces**.
  - Las **interfaces permiten mucha más flexibilidad** para conseguir que dos clases tengan el mismo comportamiento, independiente de su situación en la jerarquía de clases de Java.
  - Las **interfaces tienen una jerarquía propia**, independiente y más flexible que la de las clases, ya que tienen permitida la herencia múltiple.

Interfaces

- Todos los miembros son públicos (no hay necesidad de declararlos públicos)
- Todos los métodos son abstractos (no se requiere declararlos como abstract)
- Todos los campos datos son static y final. Se usa para definir valores constantes.
- No se permite crear objetos instancias de una Interfaz. Por la misma razón que no se puede crear instancias de clases abstractas. New InterfazX();

Interfaces - Ejemplo

- Dado el siguiente diagrama
  - ¿Qué puede comentar al respecto?
  - Revisar los código asociados