Tipos de datos abstractos

Abstract Data Types (ADT)

Tipos primitivos

- Soportados directamente por el procesador
- Operaciones aritmético/lógicas en estos tipos de datos

e.g. Tipos primitivos en Java

```
byte       float      boolean       char
short      double
int
long
```

Tipos de datos abstractos (ADT)

- Proveen abstraciones de más alto nivel
- Buscan mejorar la modularidad del código
- Se componen de valor y operaciones
- Encapsulan el valor del tipo
- Los clientes hacen uso a través de la interfaz de programación (API – Application Program Interface)

Roles respecto a un ADT

- Cliente/Usuario : Hace uso del ADT a través de su interfaz de programación (API). El ADT oculta todos los detalles de la implementación (representación, implementación de las operaciones)
- Implementador del ADT: Decide la representación del valor y los algoritmos para implementar las operaciones del ADT.

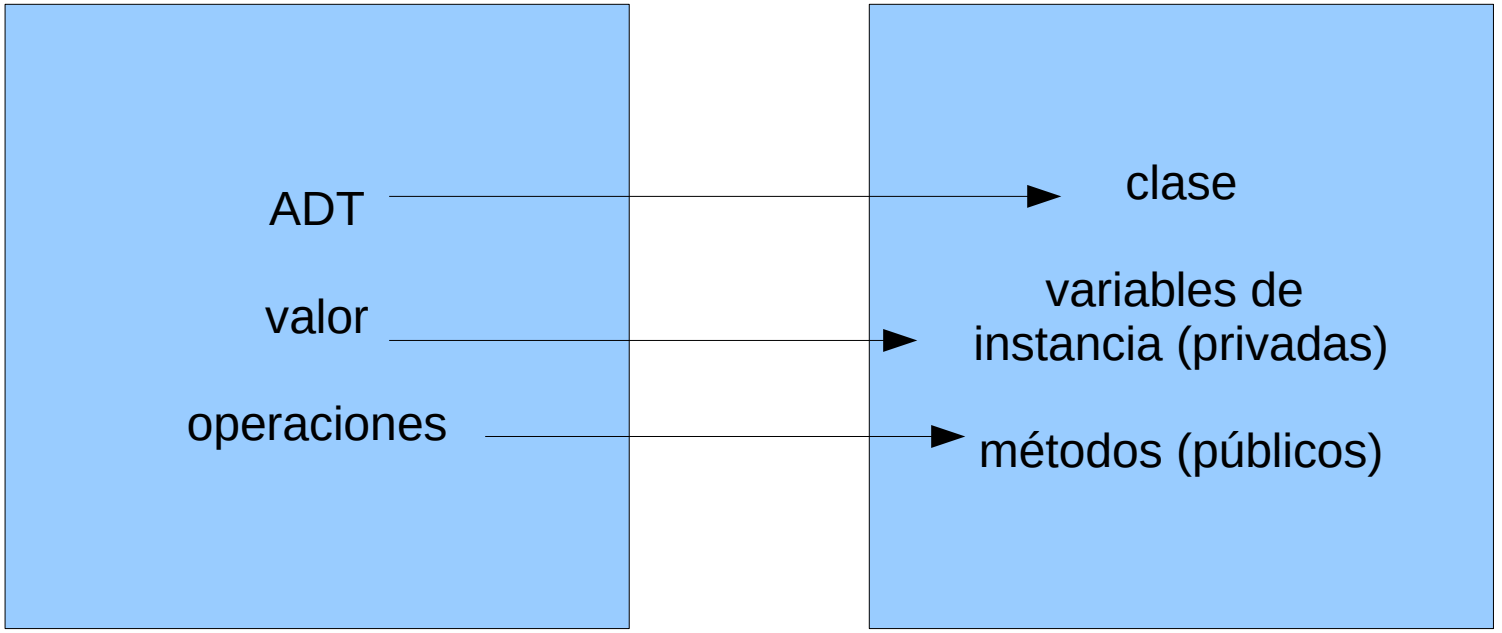
Beneficios de los ADT

- Alto nivel de abstracción: Modelan el problema a resolver. Facilitan entender y desarrollar la solución.
- Encapsulación: Solo se accede a través de su interfaz pública. Oculta representación e implementación. Garantiza el uso consistente.
- Modularidad: Fácilmente se pueden hacer cambios a la representación y la implementación del ADT sin afectar código cliente. Facilita la reutilización.

Ejemplo ADT

- El ADT Contador
- Interfaz pública
  - `void incrementar()` // incrementar el contador
  - `int obtenerConteo()` // obtener el valor
  - `Contador()` // Constructor del Contador
- Ejemplo cliente: Software de conteo de votos

Implementación de ADT



Dato del tipo ADT → Instancia de la clase: Objeto

Notas Java

- Variables de instancia privadas garantizan el encapsulamiento del valor. La representación es oculta al cliente.
- Métodos de instancia públicos constituyen el API. La implementación es oculta al cliente.
- Métodos estáticos: Bibliotecas de funciones. No están asociados al ADT pues no tienen acceso a las variables de instancia.

Ejemplos de bibliotecas

Bibliotecas del Java

- java.lang.Math
- java.util.Arrays

Texto guía

- edu.princeton.cs.algs4.StdIn
- edu.princeton.cs.algs4.StdOut
- edu.princeton.cs.algs4.StdDraw
- edu.princeton.cs.algs4.StdRandom
- edu.princeton.cs.algs4.StdStats

Método `main`

Método estático que busca la máquina virtual para ejecutar programas.

Por ejemplo

```
java paquete.Clase
```

ejecuta el `main` de *`Clase`*.

Pruebas unitarias

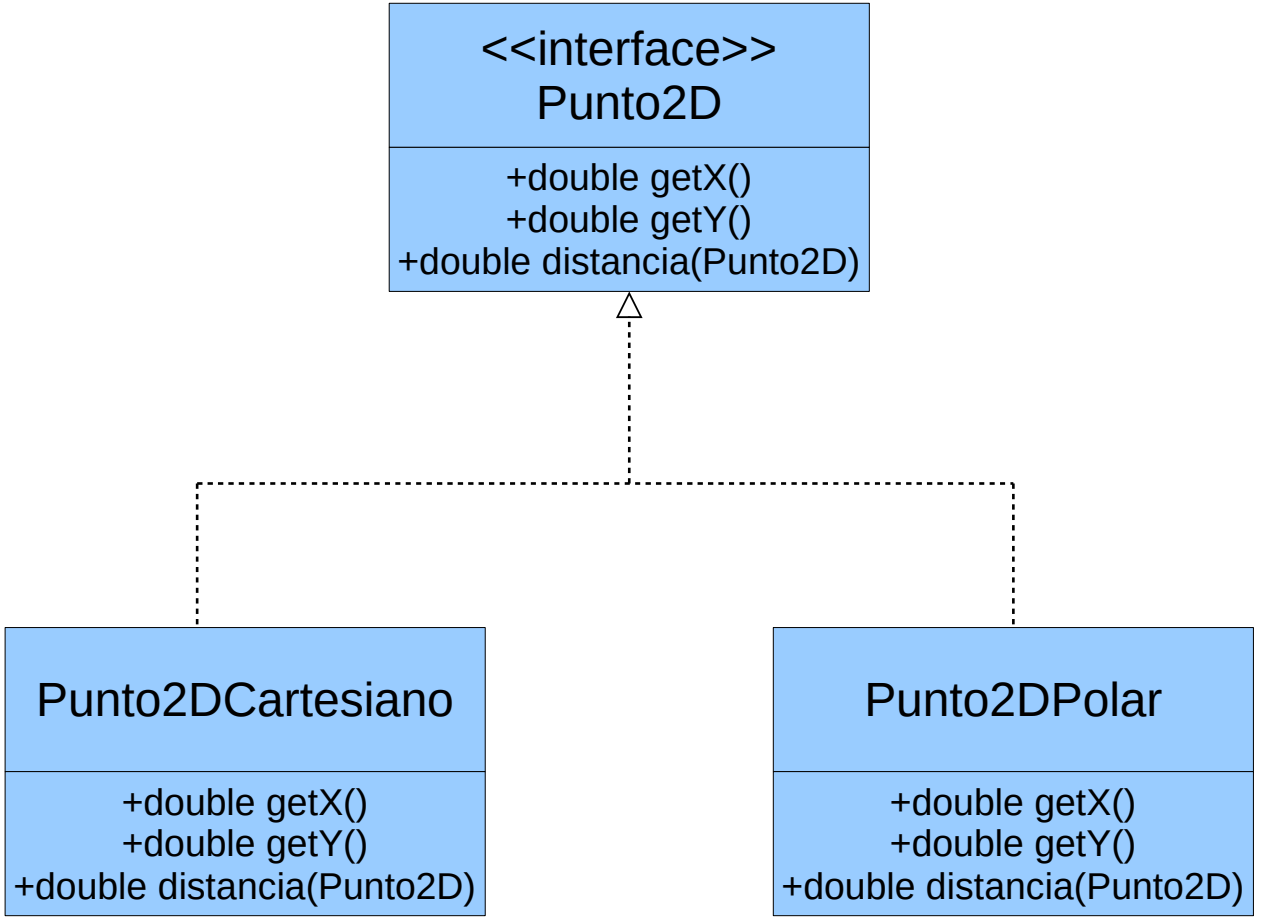
- Pruebas para verificar el correcto funcionamiento de una clase
- Se considera una *buena práctica de programación* incluir pruebas unitarias de cada clase (se pueden hacer en el `main` de cada clase).
- Existen herramientas para formalizar el proceso de pruebas unitarias, e.g. JUnit.

Herencia en Java

Hay dos tipos

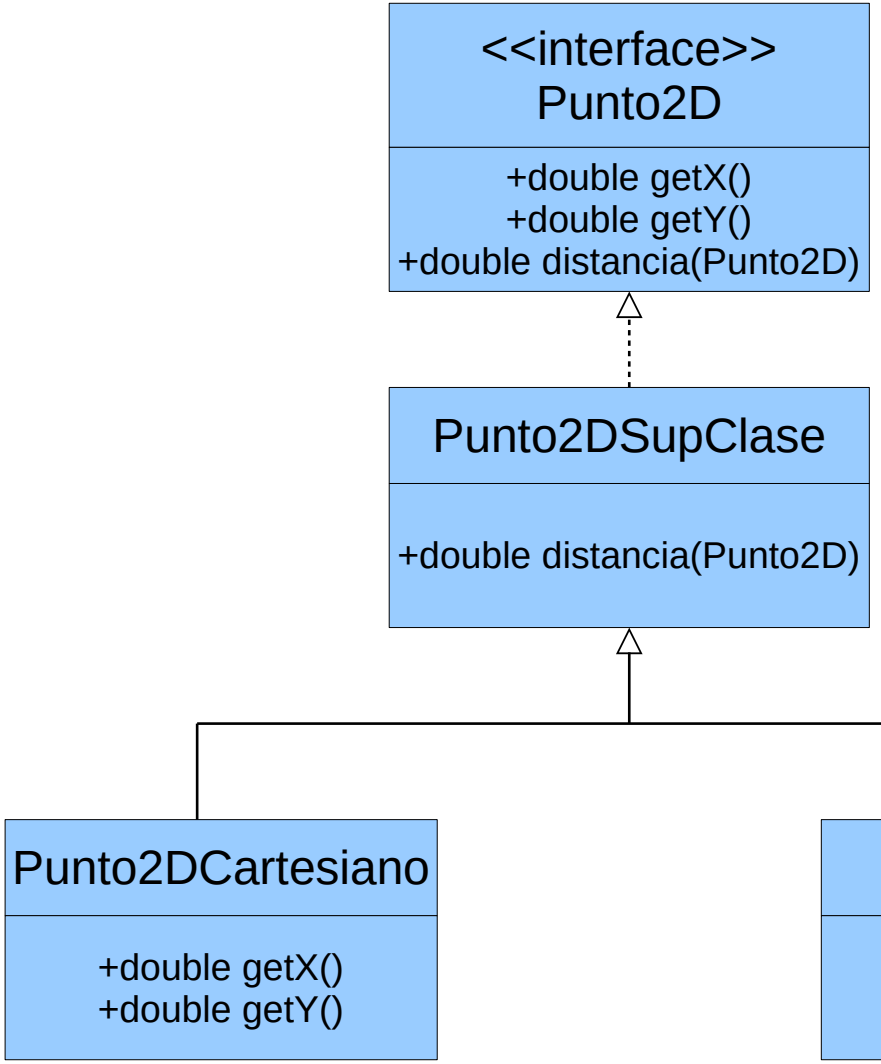
- Herencia de interfaces
  - Se logra con interfaces (`implements`).
  - Las clases que implementan la interfaz se dice que son del subtipo de la interfaz.
- Herencia de implementación
  - Se logra con la herencia (`extends`).
  - Las clases que heredan de una superclase se denominan subclases.
  - Se heredan implementaciones, pero se pueden sobre-escribir.

Ejemplo:
Herencia de Interfaces



Ver código fuente implementación

Ejemplo:
Herencia de Implementación



Superclase:
Implementación común
del método `distancia`

Herencia de Object

- Toda clase es descendiente de Object
- Algunos métodos importantes a considerar, que en muchos casos se deben sobre-escribir al implementar un ADT:
  - `boolean equals(Object x)`
    Comparar igualdad con otro objeto. Debe ser falso si el otro objeto es null o de diferente clase. Debe ser relación de equivalencia.

Relación de equivalencia
Una relación reflexiva, simétrica y transitiva

Métodos heredados de Object

- `int hashCode()`
  Retorna un código hash asociado al objeto. Debe ser consistente e idealmente objetos iguales deben retornar el mismo código.
- `String toString()`
  `Retornar una representación alfanumérica del objeto.`
- `Class getClass()`
  `Retorna la clase a la que pertenece el objeto`

Ejemplo: Sobre-escritura de los métodos heredados de Object

```
public class Punto2DCartesiano implements Punto2D {

    /**
     * Obtener una representacion textual del objeto
     */
    public String toString() {
        return "("+getX()+","+getY()+")";
    }

    /**
     * Determinar igualdad entre puntos
     */
    public boolean equals(Object a) {
        if (a==null) return false;
        if ( !(a instanceof Punto2D) ) return false;
        Punto2D px = (Punto2D) a;
        if (px.getX()==this.getX() && px.getY()==this.getY()) return true;
        return false;
    }

    ...

}
```