Classe Tipus de dades Java:


```
package meuPackage;
```
OPCIONAL Indica en quin paquet (directori) està definida la classe

```
import llibreria.*;
```
OPCIONAL Indica en quin paquet estan les classes que s'importen (també pot importar-se una única classe)

```
public class MeuaClasse {
```
Modificador de visibilitat `public`: fa que `MeuaClasse` siga visible des de qualsevol altra

Identificador de la classe; ha de començar per majúscula i la resta en minúscules, excepte noves paraules

Atributs o variables de classe.
Accés des de fora: `MeuaClasse.CONSTANT_CLASSE` o `MeuaClasse.atributDeClasse`

```
    public static final tipus CONSTANT_CLASSE = valor;
```
Indica que l'atribut NO POT SER MODIFICAT, és una constant. L'identificador ha d'estar tot en majúscules i "_" per separar paraules

Tipus de dades de l'atribut

```
    private static tipus atributDeClasse;
```

Atributs o variables d'instància (sense `static`). L'identificador ha de començar en minúscula

```
    private tipus1 atribut1;
    private tipus2 atribut2;
    ...
```

Fa que els atributs siguen visibles només des de `MeuaClasse`

Constructor general. Amb el mateix nom que la classe i tants paràmetres com atributs d'instància. No té tipus de retorn

```
    public MeuaClasse(tipus1 param1, tipus2 param2, …, tipusN paramN) {
        this.atribut1 = param1;
        this.atribut2 = param2;
        ...
        atributDeClasse = ...;
        ...
    }
```

`this` fa referència a l'objecte en curs; si no hi ha ambigüitat, el seu ús és opcional

Des de qualsevol mètode dinàmic es pot accedir a tots els atributs (d'instància i de classe)

Constructor per defecte. Amb el mateix nom que la classe i sense paràmetres. No té tipus de retorn

```
    public MeuaClasse() {
        this(arg1, arg2, ..., argN);
    }
```

`this` per a invocar a un altre constructor, el que concorde en el número, tipus i ordre dels paràmetres de la seua capçalera

Els identificadors dels mètodes no constructors han de començar en minúscula i si la paraula és composta, sense blancs i les inicials en majúscules

Instrucció `return`: tot mètode que no siga `void` ni constructor, ha de tornar un resultat del mateix tipus o compatible amb el tipus de retorn

Tipus de retorn, pot ser elemental, referència o `void`

```
    public tipus1 getAtribut1() { return atribut1; }
```

Mètodes consultors: un per cada atribut privat que s'haja de consultar des de fora. El tipus de retorn ha de coincidir amb el de l'atribut

```
    public void setAtribut1(tipus1 param1) { atribut1 = param1; }
```

Mètodes modificadors: un per cada atribut privat que s'haja de modificar des de fora. El tipus de retorn és `void`

```
    public tipus altresMetodesDInstancia(...) { ... }
```

Altres mètodes d'instància que no són `gets` ni `sets`. Recorda que els mètodes `void` no necessiten `return`

```
    public static tipus metodeEstatic(...) { ... }
```

Des d'un mètode estàtic o de classe només es pot accedir als atributs i mètodes estàtics de `MeuaClasse`

```
    public String toString() { … }
```

Sobreescriptura de mètodes d'`Object`. La capçalera no ha de modificar-se. El cos ha de contindre les instruccions apropiades per a la classe

```
    public boolean equals(Object o) {
        return (o instanceof MeuaClasse)
            && atribut1 == ((MeuaClasse) o).atribut1
            && atribut2.equals(((MeuaClasse) o).atribut2)
            && ...;
    }
}
```

`instanceof` per a comprovar que l'`Object o` és del mateix tipus que `this`

Expressió lògica que s'avaluarà a `true` sempre que l'objecte en curs (`this`) i `o` hagen de ser considerats iguals. Sol ser una comprovació d'igualtat atribut a atribut. Recorda que cal fer un casting de l'objecte `o` al tipus de la classe de `this` i comparar cada atribut amb == si és de tipus elemental o amb `equals` si és de tipus referència