# Records

# Características y definición

- > Reduce la definición de clases tipo JavaBean.
- >Se definen con la palabra record, indicando la firma del constructor:

```
public record Persona(String nombre, String email, int telefono) {
}

Persona persona=new Persona("fulanito","fulanito@gmail.com",22);
```

- >Se genera de forma automática:
  - Los atributos de la clase
  - •La implementación del constructor
  - •Métodos para recuperación de valores: nombre(), email(), telefono()
  - Implementaciones de toString() y equals()

## Consideraciones

- >Los records son implícitamente finales, por lo que no pueden ser heredados
- >Un record no puede heredar ninguna clase existente, aunque si puede implementar interfaces
- >Los atributos son inmutables, no pueden ser modificados
- >Los atributos son finales. Si se define explícitamente un constructor, se deben dar valores a todos sus atributos
- No pueden definir variables de instancia, si métodos de instancia y también variables de clase(static)

### Otros constructores

➤ Constructor compacto: Constructor especial que permite realizar validaciones y transformaciones de datos:

>Sobrecarga de constructores: Se pueden incluir constructores adicionales, pero siempre tendrán que llamar al canónico:

```
public record Punto(int x, int y) {
   public Punto(int x) {
     this(x,x);
   }
}
```

## Revisión conceptos



#### Dado el siguiente record:

```
public record Book(String title, double price){
   private final static double PLUS=10;
   //línea 1
}
```

### ¿Cuál de los siguientes bloques de código sería válido en línea 1?

- a. public Book(){this(PLUS);}b. public Book(double price){price=price\*PLUS;}
- c. public Book{price+=PLUS;}
- d. public Book(double p){this(p);}

Respuesta

La respuesta es la c. La a y d realizan de forma incorrecta la llamada al constructor canónico, mientras que el constructor definido en b no incluye dicha llamada. En la c se implementa de forma adecuada el constructor compacto para modificar uno de los parámetros.