Level 1 (Parent)

```
constructor() {
   Syso( Parent → Execute)
}
```

Level 2 (child)

```
constructor() {
      super();
      Syso(child → Execute);
}
```

Level 3 (Grandchild)

```
constructor() {
      super();
      Syso( Grandchild → Execute)
}
```

Grandchild obj = new Grandchild();

Invokam(call) | Execution Postorder

$$\begin{tikzcd}
GC() \arrow[d] & \textcircled{4} \\
C() \arrow[d] & \textcircled{3} \arrow[u] \\
Parent() \arrow[d] & \textcircled{2} \arrow[u] \\
object() & \textcircled{1} \arrow[u]
\end{tikzcd}$$

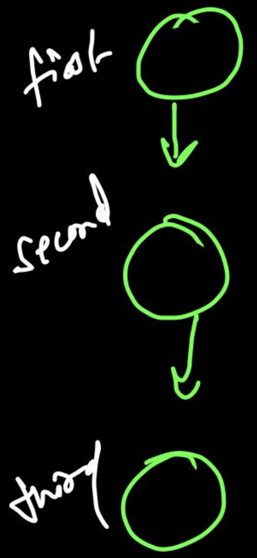
```
Level1(Parent)
finalize(){
    syso(" P→ cleanupcode");

}
Level 2 (Child)
finalize(){
    syso(" C→ cleanup code");
    super.finalize();
}

Level 3 (GrandChild)

finalize(){
    syso("GC→ cleanup code");
    super.finalize();
}
```

Grandchildobj.finalize();
{explicitly}

Invokation(call)

$$\begin{tikzcd} GC() \arrow[d] \\ C() \arrow[d] \\ Parent() \end{tikzcd}$$

Execution
(Reorder)



Forceful Execution of Object Destruction
or calling of Garbage Collection

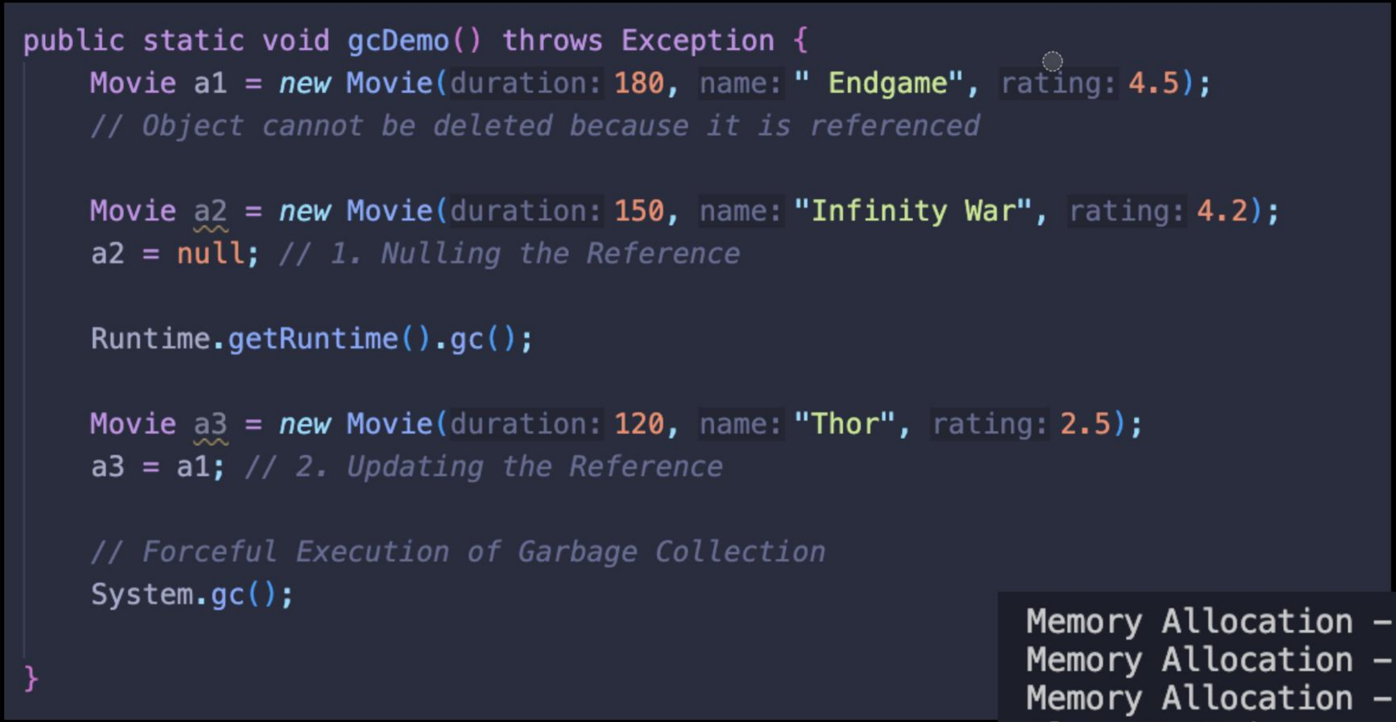
$$\begin{tikzcd}[column sep=small] \text{(1) System.gc();} \arrow[r] \arrow[dr] & \text{finalize} \arrow[r] & \text{cleanup code} \\ & \text{object destroy} & \end{tikzcd}$$

(2) Runtime.getRuntime().gc();

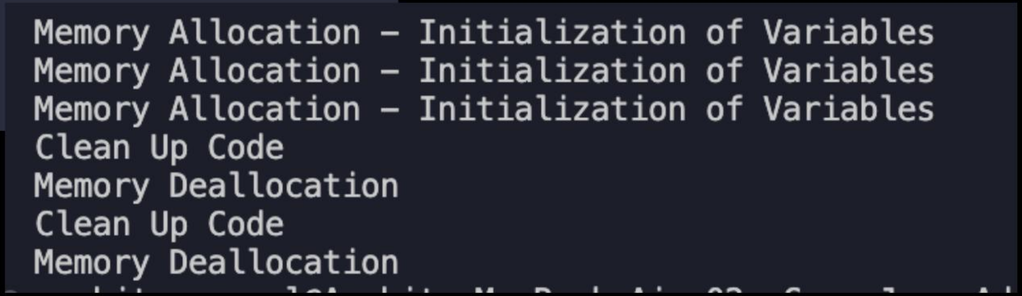
Optional for JVM to accept your GC thread. It can also refuse/decline them.

① final vs finally vs finalize

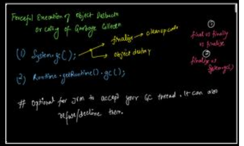
② finalize vs System.gc()



```java
public static void gcDemo() throws Exception {
    Movie a1 = new Movie(duration: 180, name: " Endgame", rating: 4.5);
    // Object cannot be deleted because it is referenced

    Movie a2 = new Movie(duration: 150, name: "Infinity War", rating: 4.2);
    a2 = null; // 1. Nulling the Reference

    Runtime.getRuntime().gc();

    Movie a3 = new Movie(duration: 120, name: "Thor", rating: 2.5);
    a3 = a1; // 2. Updating the Reference

    // Forceful Execution of Garbage Collection
    System.gc();

}
```



```
Memory Allocation - Initialization of Variables
Memory Allocation - Initialization of Variables
Memory Allocation - Initialization of Variables
Clean Up Code
Memory Deallocation
Clean Up Code
Memory Deallocation
```



Enums in Java

1) Creating interrelated constants in Java
   - old way :→ Interfaces or abstract class
   - new way :→ Enums
     - → Outside any class
     - → Inside class (inner enum)

2) Variables are → public, final, static, objects of same type
   - Internal implementatn of Enums → class

3) Constructors
   - → Empty parameter/default Constructor
   - → Parameterized Constructor

✓4) Getters & other functions can be there

✓5) ENUM parent class
- ↳ Extended by every userdefined enum.
- ↳ Implements Comparable & Serialize
- ↳ ordinal(), values(), name()

✓6) Enum cannot extend any other class
But Enum can implement other interfaces

Applications
→ switch case
→ for each loop over enum!

Eg Levels,
Days,
Colors,
Languages,
Movie Type, etc

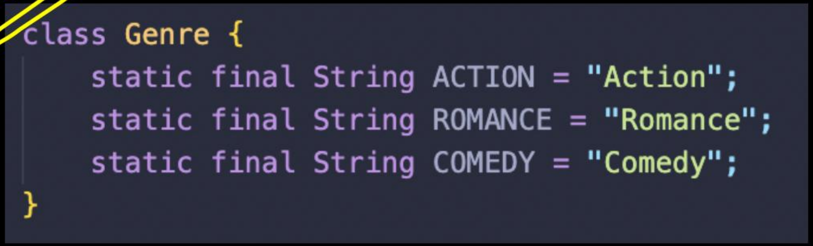
Enums in Java

1) Creating interrelated constants in Java
- ↳ old way :→ Interfaces or abstract class
- ↳ new way :→ Enums < outside any class / inside class (inner enum)

2) Variables are → public, final, static, object of same type
↳ internal implementation of Enums → class

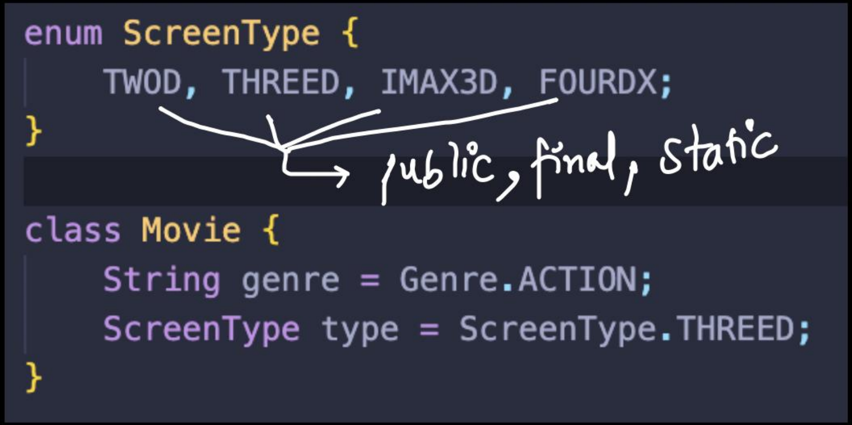
3) Constructors < Empty parameter/default constructor / Parameterized constructor

abstract / interface



```java
class Genre {
    static final String ACTION = "Action";
    static final String ROMANCE = "Romance";
    static final String COMEDY = "Comedy";
}
```

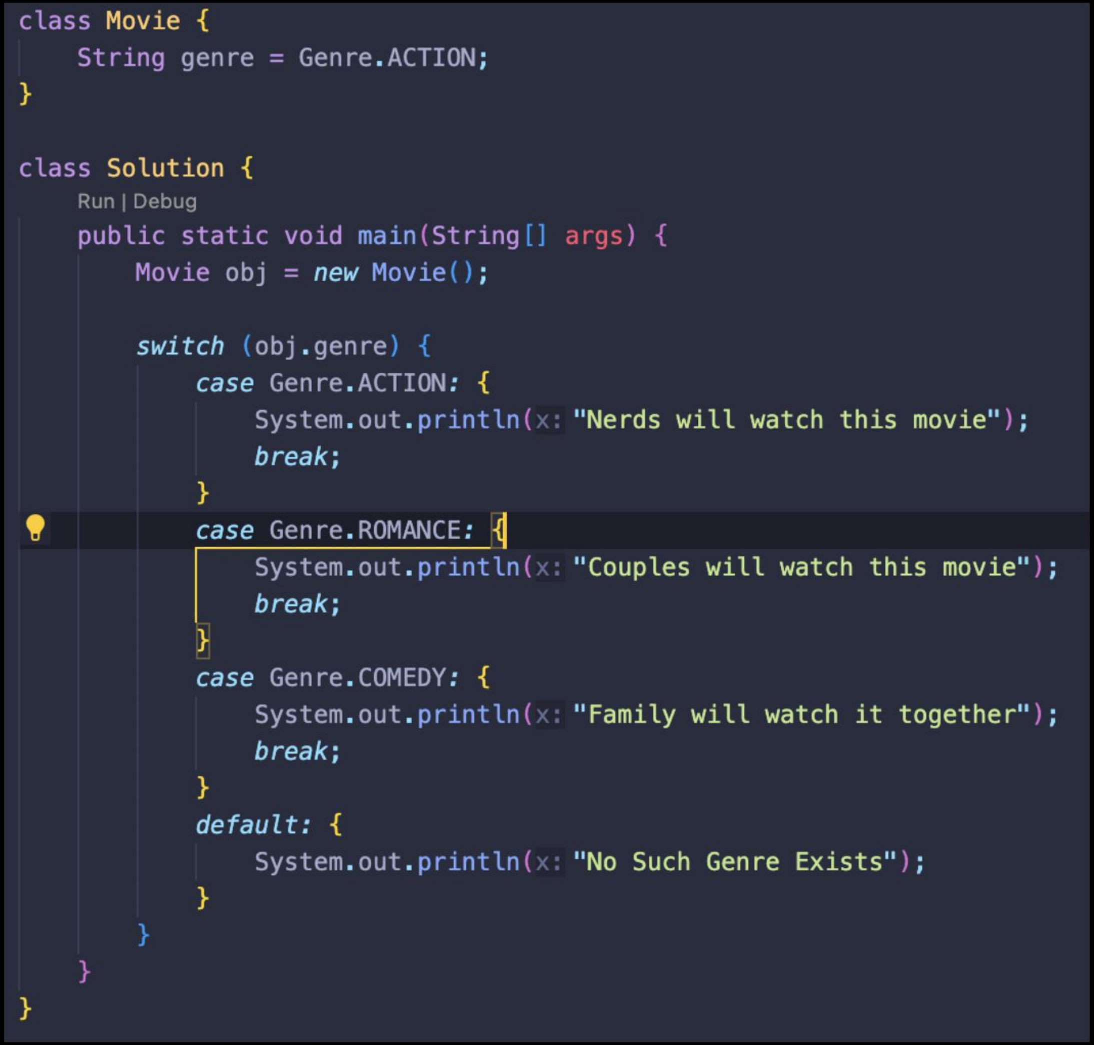
old fashioned way



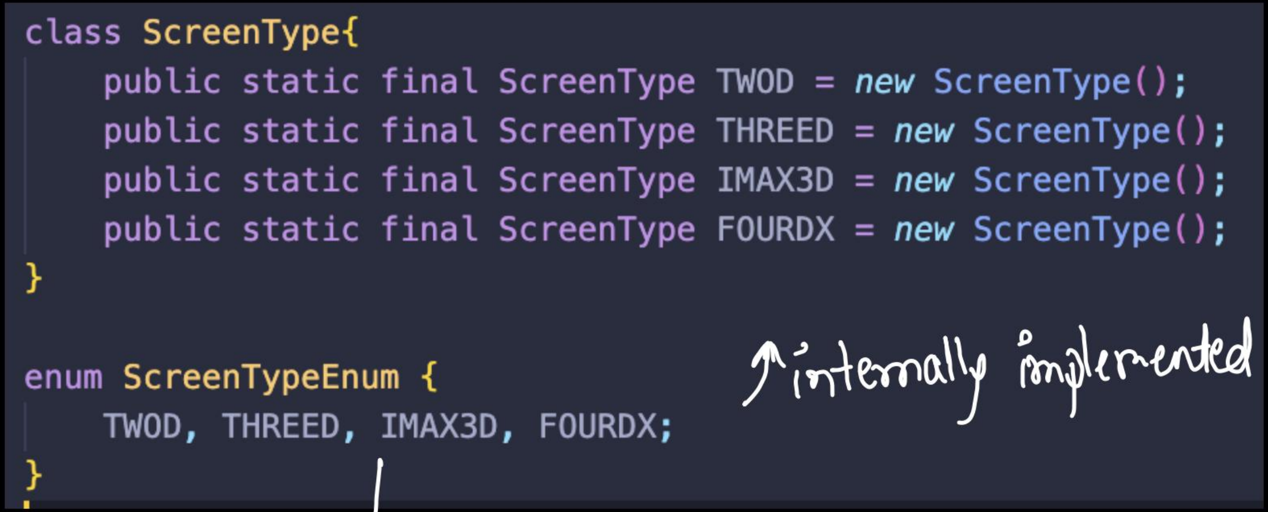
```java
enum ScreenType {
    TWOD, THREED, IMAX3D, FOURDX;
}

class Movie {
    String genre = Genre.ACTION;
    ScreenType type = ScreenType.THREED;
}
```

public, final, static

New way



```java
class Movie {
    String genre = Genre.ACTION;
}

class Solution {
    Run | Debug
    public static void main(String[] args) {
        Movie obj = new Movie();

        switch (obj.genre) {
            case Genre.ACTION: {
                System.out.println(x: "Nerds will watch this movie");
                break;
            }
            case Genre.ROMANCE: {
                System.out.println(x: "Couples will watch this movie");
                break;
            }
            case Genre.COMEDY: {
                System.out.println(x: "Family will watch it together");
                break;
            }
            default: {
                System.out.println(x: "No Such Genre Exists");
            }
        }
    }
}
```



```
class ScreenType{
    public static final ScreenType TWOD = new ScreenType();
    public static final ScreenType THREED = new ScreenType();
    public static final ScreenType IMAX3D = new ScreenType();
    public static final ScreenType FOURDX = new ScreenType();
}

enum ScreenTypeEnum {
    TWOD, THREED, IMAX3D, FOURDX;
}
```

↑ internally implemented

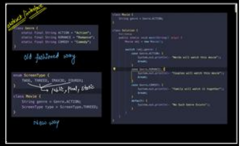
→ extends Enum

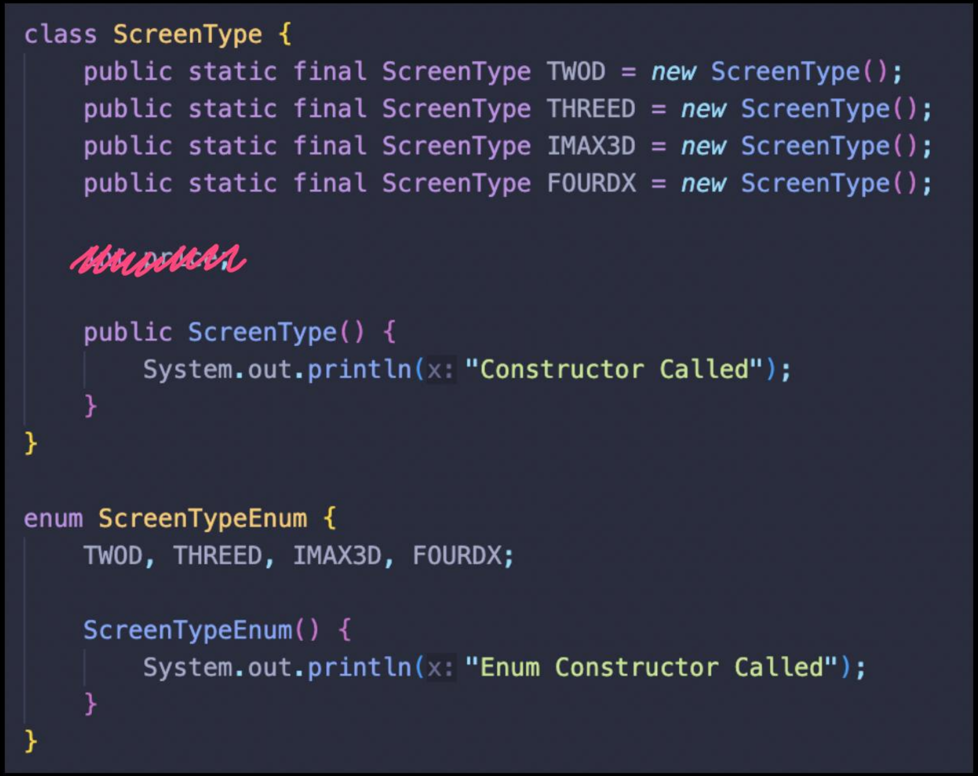
↳ cannot extend any other class

→ Although Interfaces can be implemented

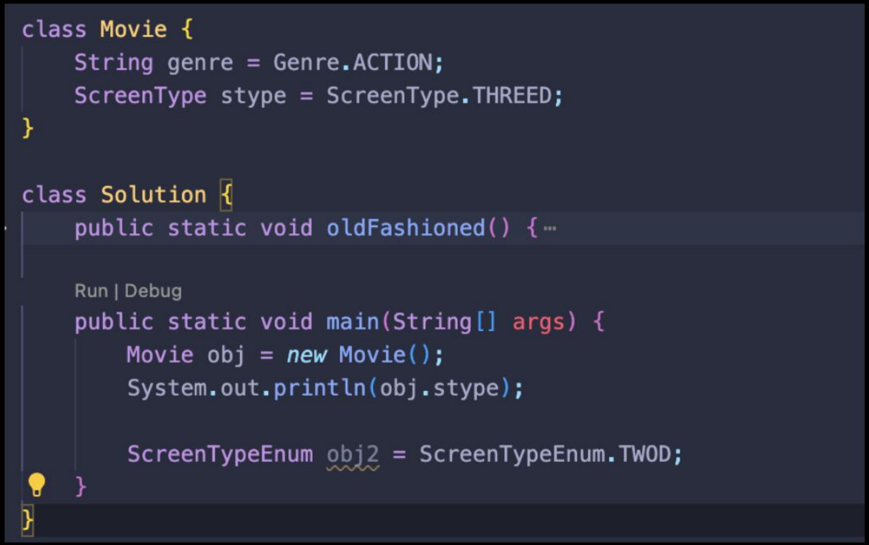
enum extends T ✗

enum implements T ✓

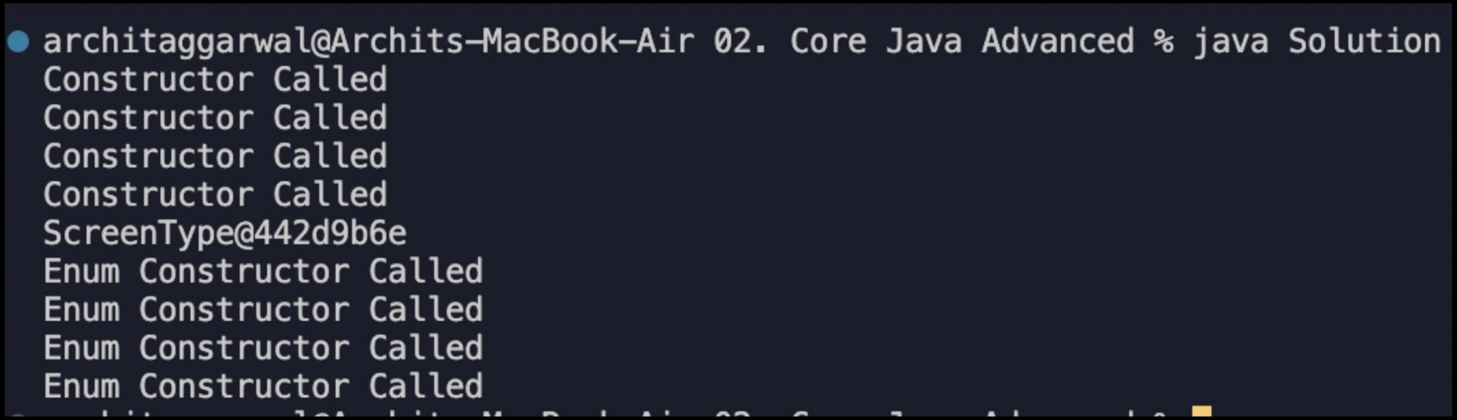




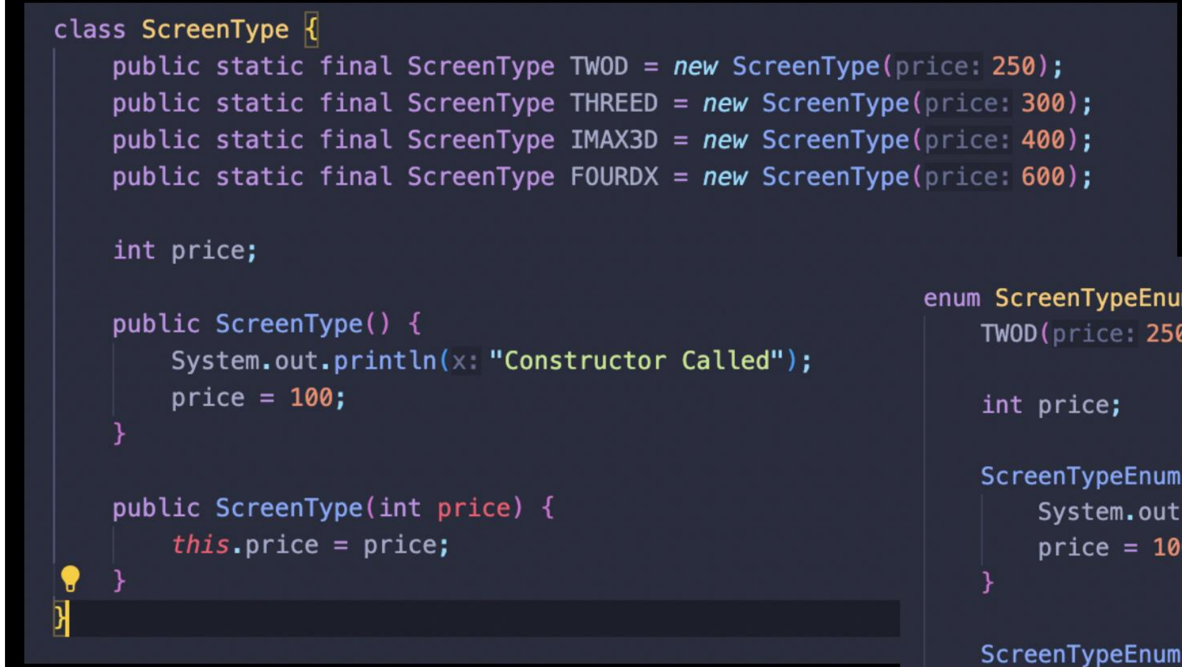
```java
class ScreenType {
    public static final ScreenType TWOD = new ScreenType();
    public static final ScreenType THREED = new ScreenType();
    public static final ScreenType IMAX3D = new ScreenType();
    public static final ScreenType FOURDX = new ScreenType();

    ~~[illegible]~~

    public ScreenType() {
        System.out.println(x: "Constructor Called");
    }
}

enum ScreenTypeEnum {
    TWOD, THREED, IMAX3D, FOURDX;

    ScreenTypeEnum() {
        System.out.println(x: "Enum Constructor Called");
    }
}
```



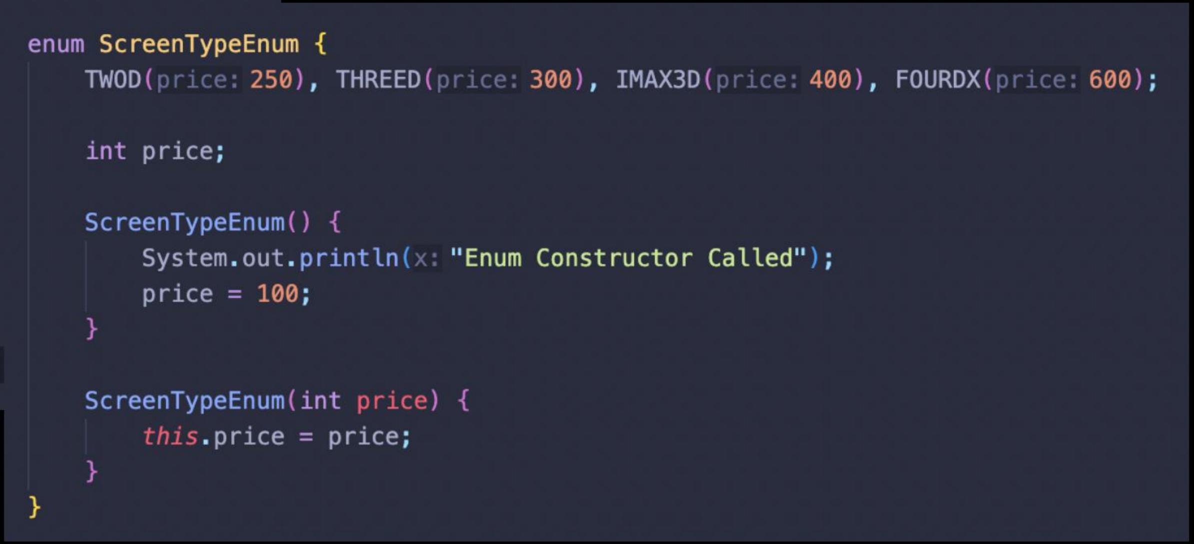
```java
class Movie {
    String genre = Genre.ACTION;
    ScreenType stype = ScreenType.THREED;
}

class Solution {
    public static void oldFashioned() {…

    Run | Debug
    public static void main(String[] args) {
        Movie obj = new Movie();
        System.out.println(obj.stype);

        ScreenTypeEnum obj2 = ScreenTypeEnum.TWOD;
    }
}
```



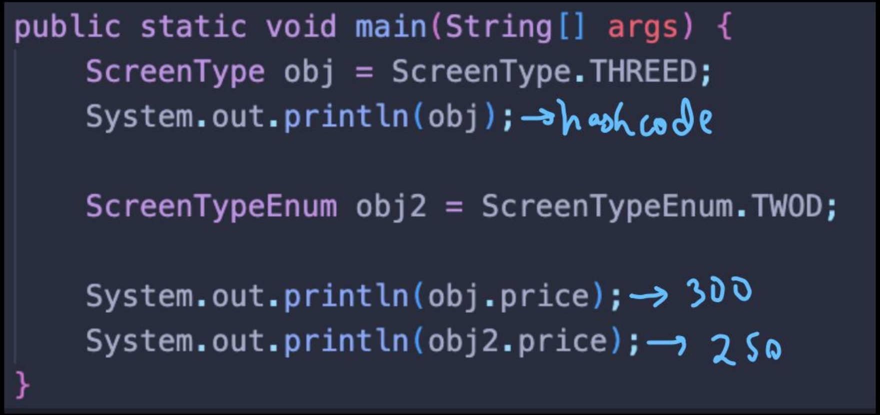
```
architaggarwal@Archits-MacBook-Air 02. Core Java Advanced % java Solution
Constructor Called
Constructor Called
Constructor Called
Constructor Called
ScreenType@442d9b6e
Enum Constructor Called
Enum Constructor Called
Enum Constructor Called
Enum Constructor Called
```



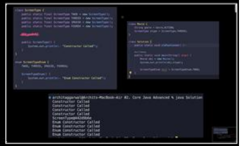
```java
class ScreenType {
    public static final ScreenType TWOD = new ScreenType(price: 250);
    public static final ScreenType THREED = new ScreenType(price: 300);
    public static final ScreenType IMAX3D = new ScreenType(price: 400);
    public static final ScreenType FOURDX = new ScreenType(price: 600);

    int price;

    public ScreenType() {
        System.out.println(x: "Constructor Called");
        price = 100;
    }

    public ScreenType(int price) {
        this.price = price;
    }
}
```

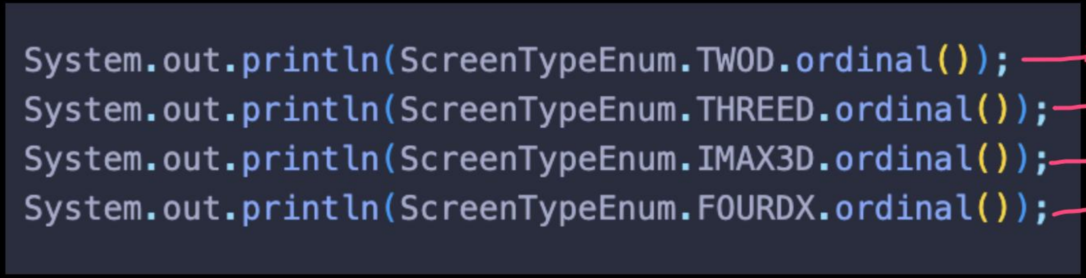


```java
enum ScreenTypeEnum {
    TWOD(price: 250), THREED(price: 300), IMAX3D(price: 400), FOURDX(price: 600);

    int price;

    ScreenTypeEnum() {
        System.out.println(x: "Enum Constructor Called");
        price = 100;
    }

    ScreenTypeEnum(int price) {
        this.price = price;
    }
}
```



```java
public static void main(String[] args) {
    ScreenType obj = ScreenType.THREED;
    System.out.println(obj);→hashcode

    ScreenTypeEnum obj2 = ScreenTypeEnum.TWOD;

    System.out.println(obj.price);→ 300
    System.out.println(obj2.price);→ 250
}
```





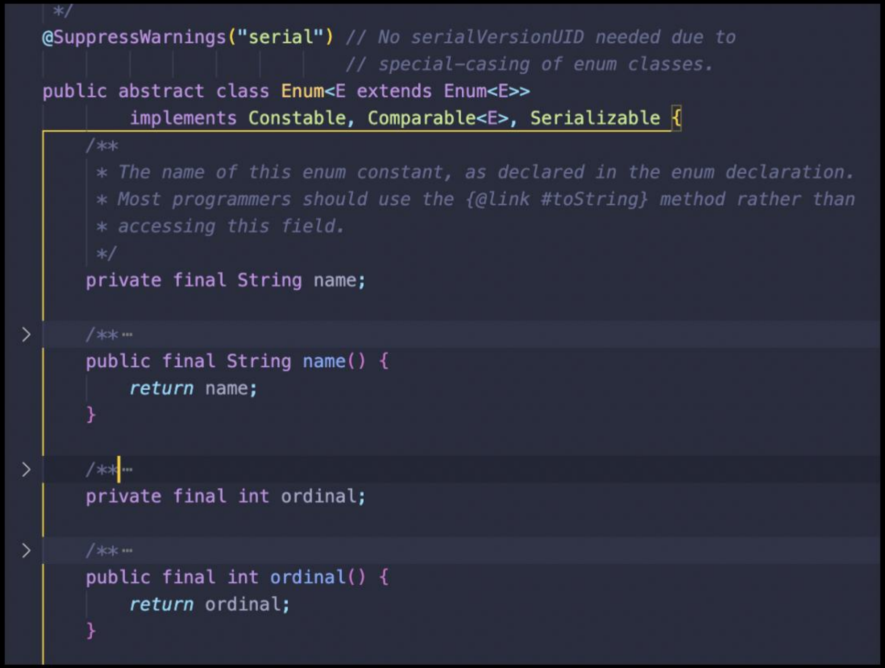
```java
System.out.println(ScreenTypeEnum.TWOD.ordinal());
System.out.println(ScreenTypeEnum.THREED.ordinal());
System.out.println(ScreenTypeEnum.IMAX3D.ordinal());
System.out.println(ScreenTypeEnum.FOURDX.ordinal());
```

→ 0
→ 1
→ 2
→ 3

ordinal property → Enum class (Parent class)

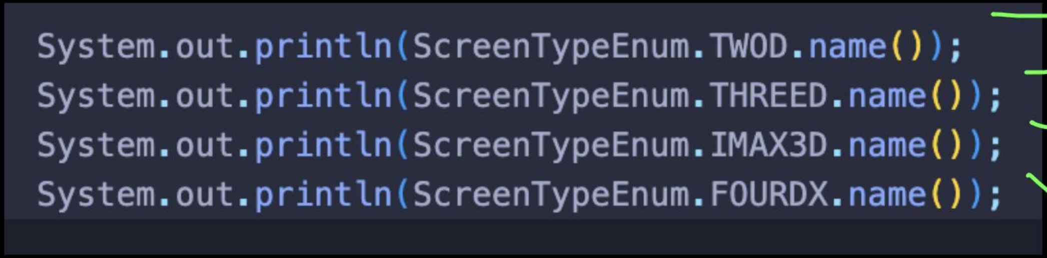
of enum

↳ to uniquely identify each constant



```java
 */
@SuppressWarnings("serial") // No serialVersionUID needed due to
                            // special-casing of enum classes.
public abstract class Enum<E extends Enum<E>>
        implements Constable, Comparable<E>, Serializable {
    /**
     * The name of this enum constant, as declared in the enum declaration.
     * Most programmers should use the {@link #toString} method rather than
     * accessing this field.
     */
    private final String name;

    /**...
    public final String name() {
        return name;
    }

    /**...
    private final int ordinal;

    /**...
    public final int ordinal() {
        return ordinal;
    }
```

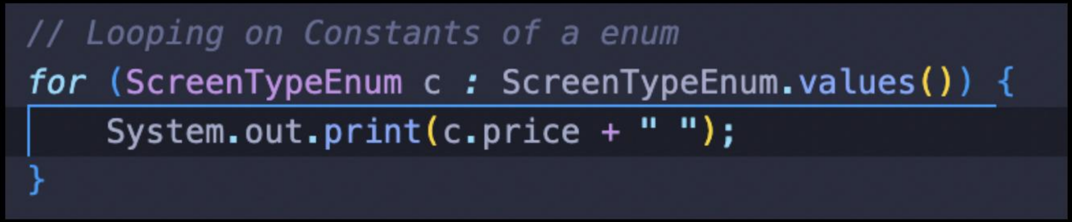
String
name



```
System.out.println(ScreenTypeEnum.TWOD.name());
System.out.println(ScreenTypeEnum.THREED.name());
System.out.println(ScreenTypeEnum.IMAX3D.name());
System.out.println(ScreenTypeEnum.FOURDX.name());
```

→ TWOD
→ THREED
→ IMAX3D
→ FOURDX

values()   List<ScreenTypeEnum>



```
// Looping on Constants of a enum
for (ScreenTypeEnum c : ScreenTypeEnum.values()) {
    System.out.print(c.price + " ");
}
```

$\}\rightarrow$ 250   300   400   600