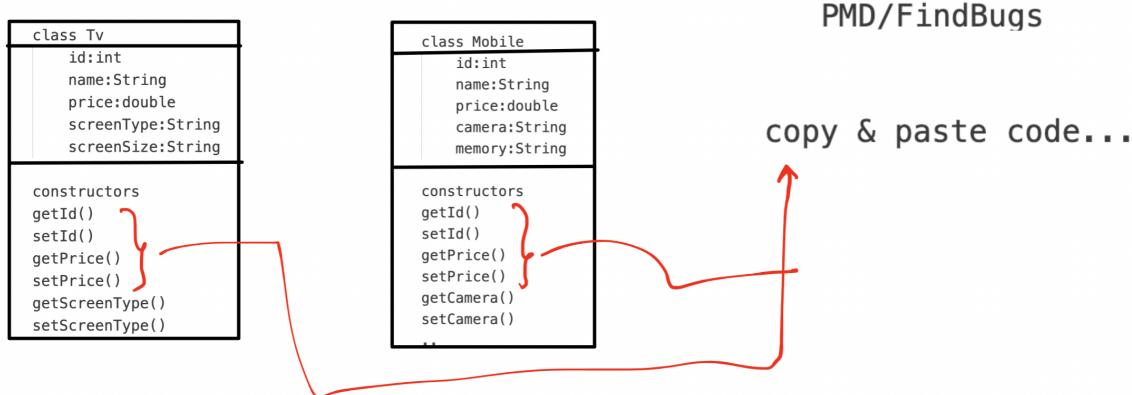
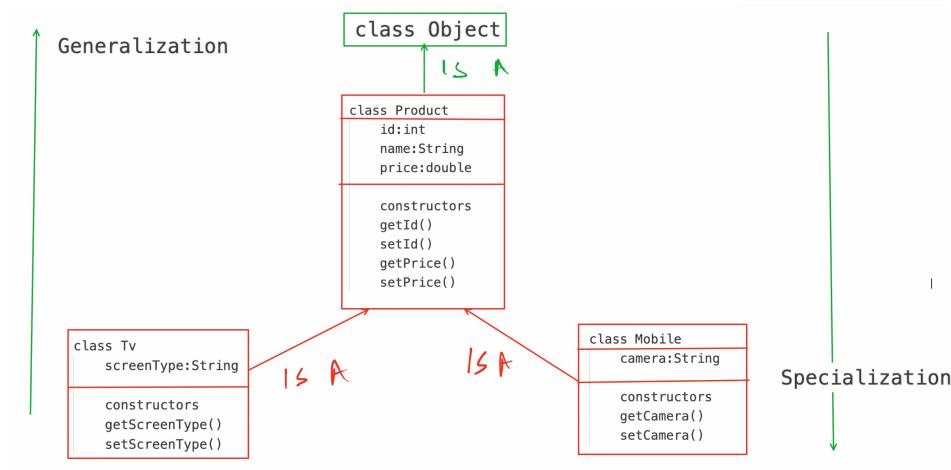


Problem: Copy and Paste



Solution: Code reusability using Generalization and Specialization relationship
Generalization and Specialization is done using **Inheritance** in programming



```

class Product {
    private int id;
    private String name;
    private double price;
}

public class Mobile extends Product{
    private String connectivity;
}

new Mobile(); // check PDF
  
```

A green box highlights the `Mobile` class definition. A green arrow points from the `Mobile` class in the UML diagram to this highlighted code.

id	name	price	connectivity
----	------	-------	--------------

```

class Product {
    public Product() {
        prints P1
    }

    public Product(int id, String name) {
        prints P2
    }
}

public class Mobile extends Product{
    public Mobile() {
        prints M1
    }

    public Mobile(int id, String name, String connectivity) {
        prints M2
    }
}

new Mobile(); // P1, followed by M1
new Mobile(34, "iPhone 6e", "5G"); // P1, followed by M1

```

Diagram annotations:

- (1) A green arrow points from the word "Object" to the class definition of "Product".
- (2) A green arrow points from the constructor of "Product" to the first line of code.
- (3) Two green arrows point from the constructor of "Mobile" to the first and second lines of code respectively.

```

class Product {
    public Product() {
        prints P1
    }

    public Product(int id, String name) {
        prints P2
    }
}

public class Mobile extends Product{
    public Mobile() {
        prints M1
    }

    public Mobile(int id, String name, String connectivity) {
        super(id, name);
        prints M2
    }
}

new Mobile(); // P1, followed by M1
new Mobile(34, "iPhone 6e", "5G"); // P2 followed by M2

```

Diagram annotations:

- A blue circle highlights the entire class definition of "Product".
- A blue arrow points from the constructor of "Mobile" to the first line of code.
- A blue arrow points from the constructor of "Mobile" to the second line of code.
- A blue arrow points from the line "new Mobile(34, "iPhone 6e", "5G");" to the text "P2".

```

class Product {
    public String getName() {
        return name;
    }

    public double getPrice() {
        return 0;
    }
}

public class Mobile extends Product{
    public String getConnectivity() {
        return "3G";
    }

    public double getPrice() { // override
        return 50,000;
    }
}

```

Diagram annotations:

- Red brackets on the left side of the code block group the "Product" class definition.
- Blue brackets on the right side of the code block group the "Mobile" class definition.
- A blue arrow points from the "getName()" method of "Product" to the "p.getName();" line in the code snippet.
- A blue arrow points from the "getPrice()" method of "Product" to the "p.getPrice();" line in the code snippet.
- A red arrow points from the "getConnectivity()" method of "Mobile" to the "p.getConnectivity();" line in the code snippet.
- The code snippet shows:
 - `p.getName(); // valid` (green checkmark)
 - `p.getPrice(); // dynamic binding --> 50,000.00 ✓` (green checkmark)
 - `p.getConnectivity(); // ✗` (red X)
- Two blue boxes at the bottom represent objects. A green arrow points from the "Mobile" class to the left box, and another green arrow points from the "Mobile" object to the right box. A red box labeled "Remote" is positioned between them.

```
products[0] = new Mobile( id: 52, name: "iPhone 6", price: 89000.00, connectivity: "5G");
```

Mobile.class	hashCode	lock	...
52	iPhone 6	89000	5G

HEADERS

