Abstract Classes & Methods

IN JAVA

Introduction :-

Imagine you're building a city of robots, each with unique abilities. Abstract classes are like the master blueprints that outline the essential features and behaviors each robot should have. Abstract methods, on the other hand, are the specific tasks these robots need to perform – think of them as task lists.

Syntax of Abstract Classes & Methods

Creating an abstract class goes like this:

```
abstract class
Robot {
    abstract void
performTask();
}
```

Here's the breakdown:

1. abstract: Signals that this class is a blueprint.
2. class: Declares that we're defining a new class.
3. Robot: Name of our abstract class.
4. abstract void performTask(): An abstract method without a body, just a task description.

Abstract Methods in Action

Now, let's create two robot types: CleanerRobot and DancerRobot.

```
class CleanerRobot extends Robot {
    @Override
    void performTask() {
        System.out.println("Cleaning up the mess.");
    }
}

class DancerRobot extends Robot {
    @Override
    void performTask() {
        System.out.println("Bust a move on the dance floor.");
    }
}
```

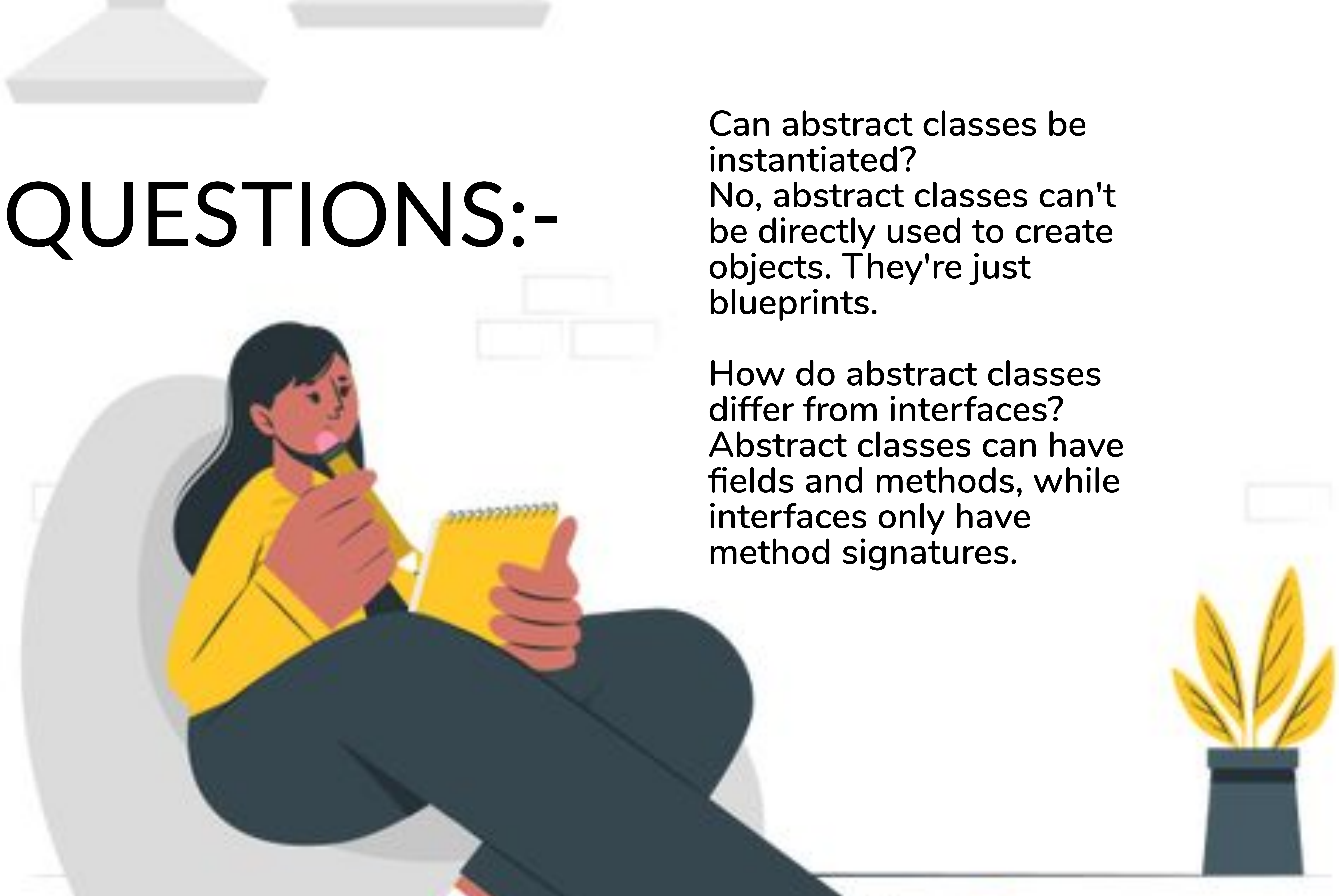
EXPLANTATION

CleanerRobot and DancerRobot extend Robot, inheriting the performTask() method.
We implement the abstract method differently for each robot type.

Advantages : -

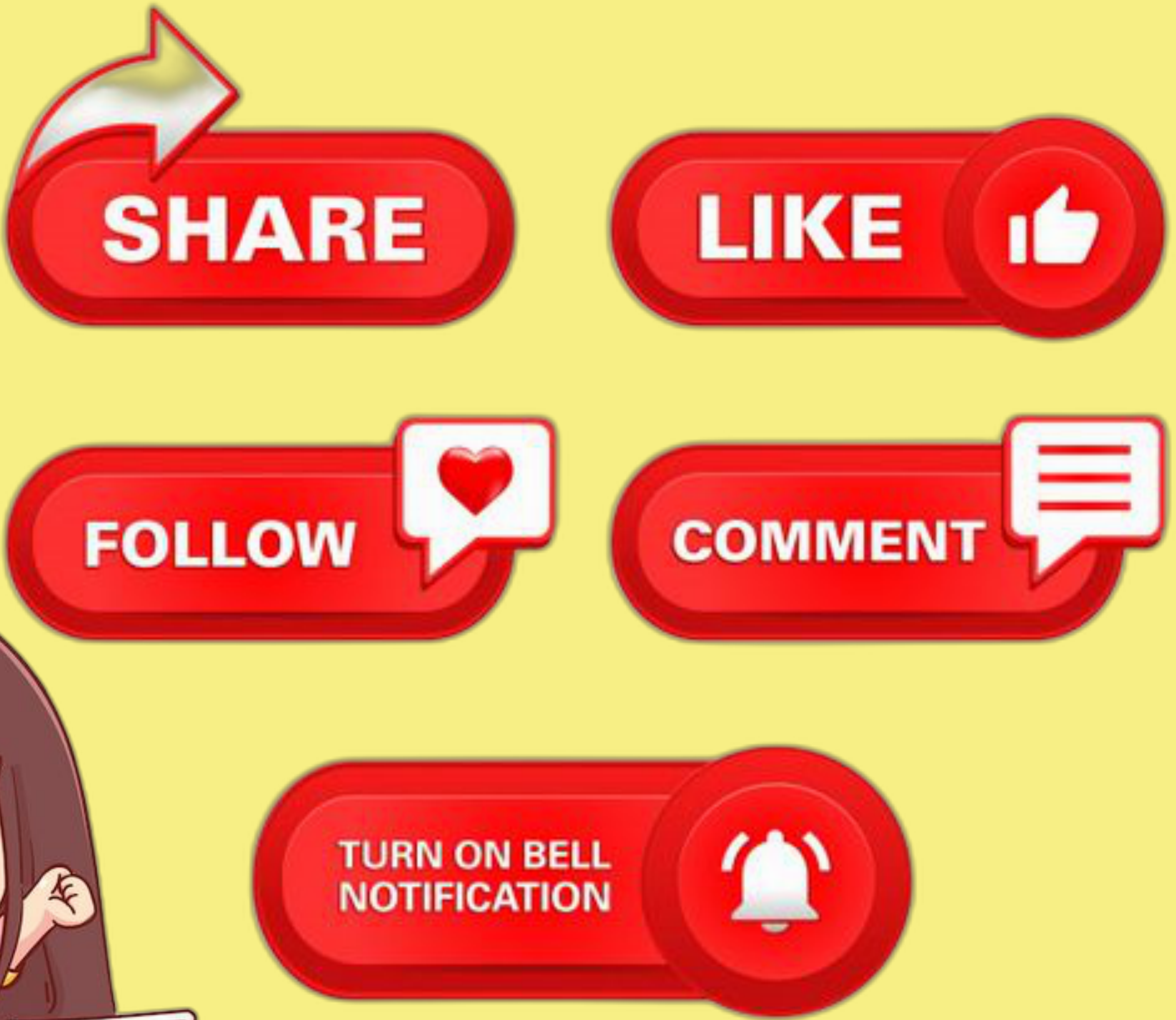
1. Consistency: Abstract classes ensure all subclasses have the same structure, making your code organized and maintainable.
2. Polymorphism: Abstract methods let you use the same method name in different classes, enhancing code flexibility.
3. Forces Implementation: Abstract methods enforce subclasses to provide specific functionality, reducing errors.
4. Efficiency: Abstract classes minimize repetitive code, saving you time and effort.
5. Design Patterns: Abstract classes play a crucial role in design patterns, helping you create efficient and scalable code architectures.

QUESTIONS:-



Can abstract classes be instantiated?
No, abstract classes can't be directly used to create objects. They're just blueprints.

How do abstract classes differ from interfaces?
Abstract classes can have fields and methods, while interfaces only have method signatures.


SHARE
LIKE
FOLLOW
COMMENT
TURN ON BELL NOTIFICATION