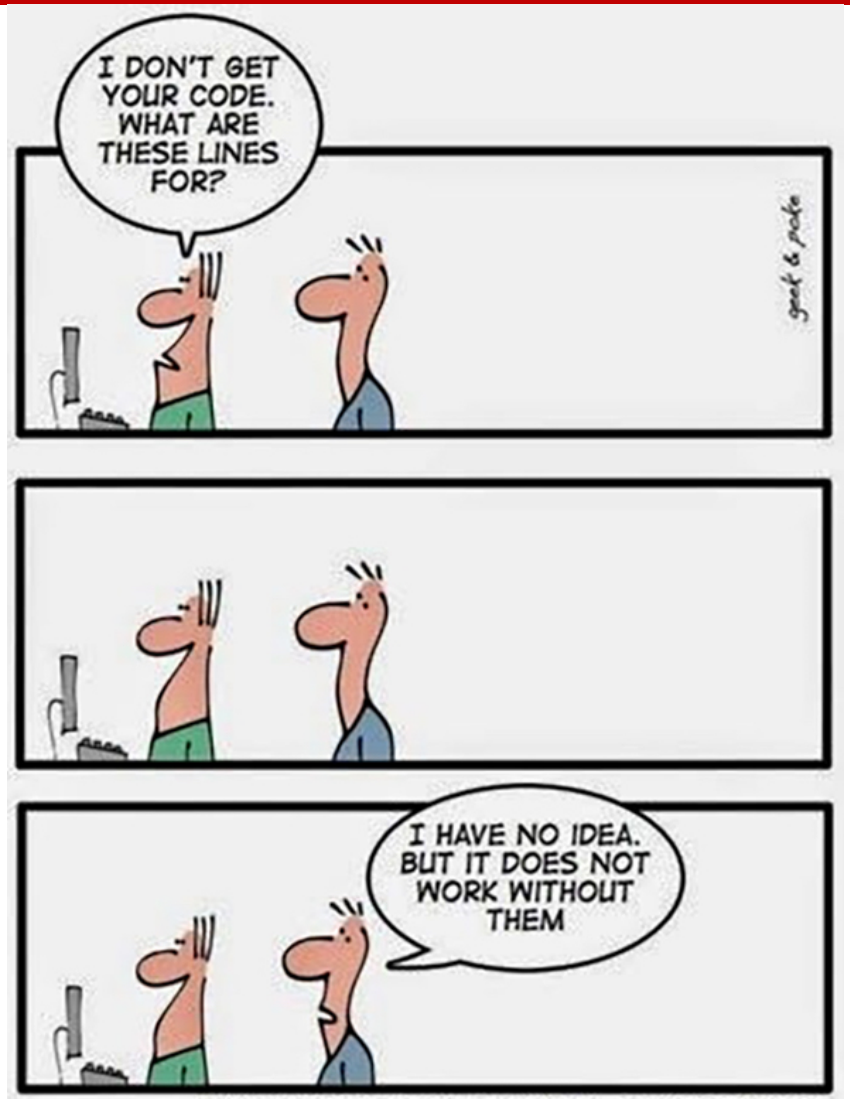
PROGRAMMING

Lecture 4

Sushil Paudel

TILL NOW


I DON'T GET YOUR CODE. WHAT ARE THESE LINES FOR?
I HAVE NO IDEA. BUT IT DOES NOT WORK WITHOUT THEM

PREVIOUS TOPICS

- Control Statements – If Else
- Control Statement – Switch
- Control Statement Loop – For, While, Do-While
- Break and Continue

TODAY'S TOPIC

- Object-Oriented Programming (OOPs)
- Class
- Object
- Methods

OBJECT-ORIENTED PROGRAMMING

- Object-Orientation in computing was invented so that doing programming would be similar to real world objects.
- Look around right now and you'll find many examples of real-world objects: your dog, your desk, your television set, your laptop, etc.
- Real-world objects have state and behavior.
- Dogs have state (name, color, breed, etc.) and behavior (barking, fetching, eating, etc.).
- Bicycles also have state (current gear, color, model, etc.) and behavior (changing gear, applying brakes, etc.).
- Identifying the state and behavior for real-world objects is a great way to begin thinking in terms of object-oriented programming.

OBJECT ORIENTED PROGRAMMING (OOPS)

- An object-based application in Java is based on declaring classes, creating objects from them and interacting between these objects.
- In computing, a date can be an object, a bank account can be an object etc.
  - Each bank account has an identity.
  - Each bank account has a balance, an account holder's name, a creation date, etc.(attribute)
  - Each bank account has some behaviors, it can be opened, closed, deposited, withdrawn, etc.

CLASS

- A class is a blueprint, a detailed description, a definition.
- A class describes what an object will be.
- Object oriented programming is not only about object but also about classes.
- We use classes to create objects.

CLASS

- To make a house, we make the blueprint first.
- It describes everything about how the house will be built.
- But the blueprint is not the house.
- We write the class first (as a blueprint) and use that class to create the object.
- We can create thousands of objects using the single class.
- So, class comes first before objects.

BLUEPRINT

CREATING A CLASS

- **Name** (Type): *what is it?*
  - Employee, BankAccount, Event, Document, Album
- **Attributes/State** (properties, data): *what describes it?*
  - width, height, color, length, filetype
- **Behavior** (operations, methods): *what can it do?*
  - play, bark, sing, sit, run, deposit, withdraw, create, delete

CREATING A BANK ACCOUNT CLASS

- **Name**: BankAccount
- **Attributes**: accountNumber, balance, dateOpened, accountType
- **Behavior**: open(), close(), deposit(), withdraw()

CLASS SYNTAX

Syntax:

```
Modifiername class className{
        // Codes
}
```

Example

```
public class College {


}
```

OBJECT

- An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc.
- An object has three characteristics:
  - **State:** represents the data (value) of an object
  - **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
  - **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

OBJECT

- **An object is an instance of a class.** A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

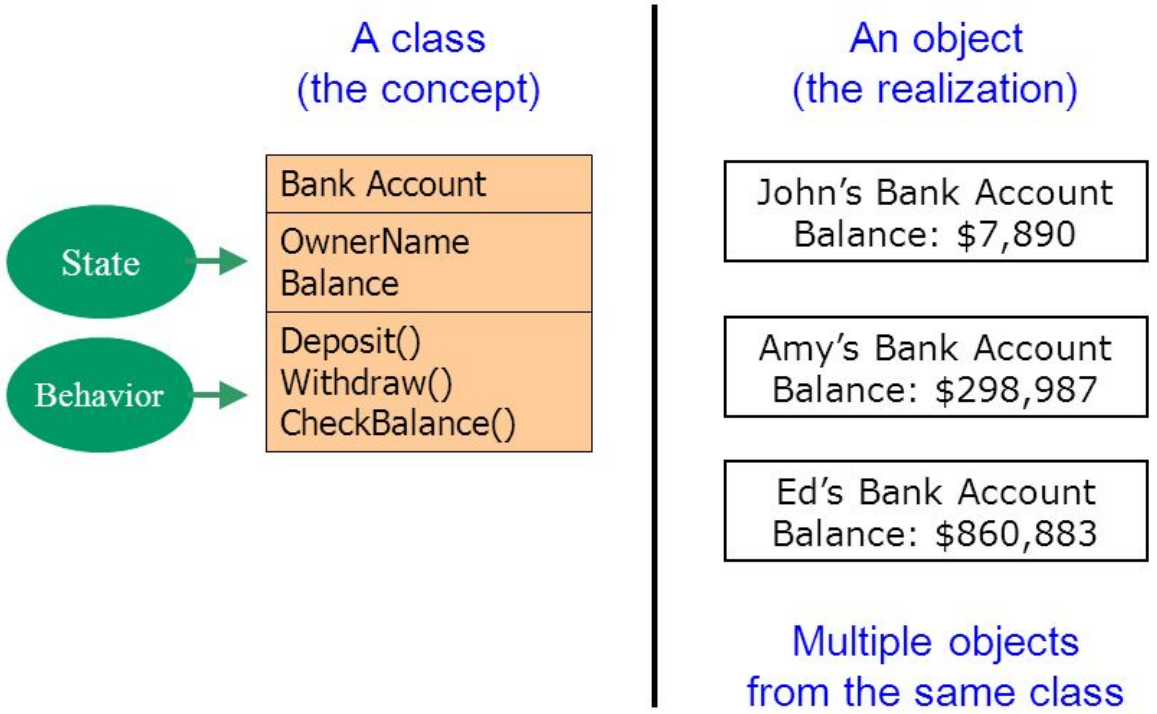
- For Example, Pen is an object. Its name is Cello, color is black, known as its **state**. It is used to write, so writing is its **behavior**.

STATE

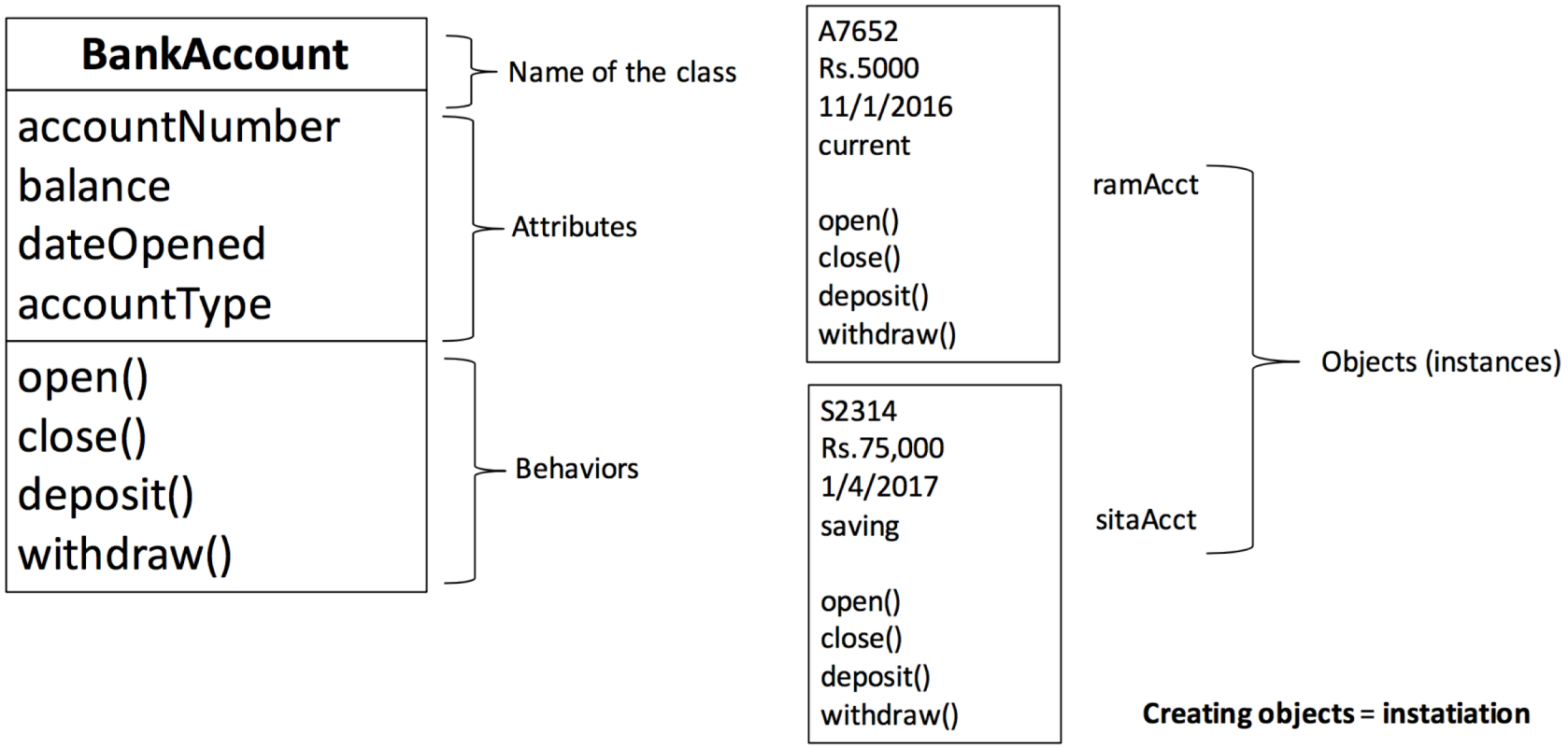
BEHAVIOR

CLASS AND OBJECT

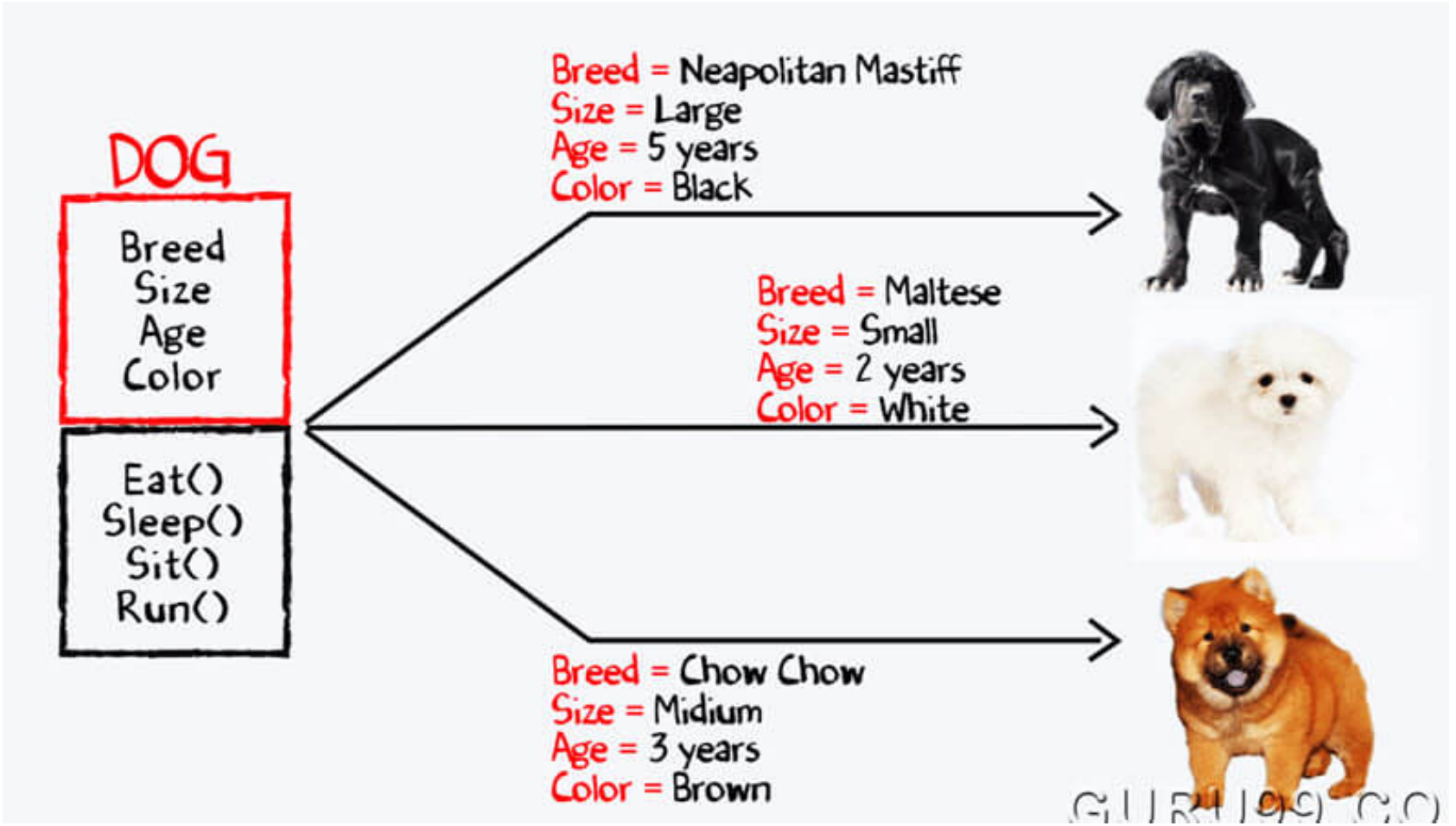
Example



CLASS AND OBJECT



DOG


DOG
Breed
Size
Age
Color
Eat()
Sleep()
Sit()
Run()
Breed = Neapolitan Mastiff
Size = Large
Age = 5 years
Color = Black
Breed = Maltese
Size = Small
Age = 2 years
Color = White
Breed = Chow Chow
Size = Midium
Age = 3 years
Color = Brown

DOG

CREATE OBJECT

- **Syntax:**

  ClassName objectName = **new** ClassName();

- **Example:**

  Animal animal = **new** Animal();

  int a= 10;

- **new** keyword is used to create an instance of the class. In other words, it instantiates a class by allocating memory for a new object and returning a reference to that memory.
- We can create any number of objects from single class using new keyboard.

METHODS

- A method is a collection of statements that perform some specific task and return the result to the caller.
- A method can perform some specific task without returning anything.
- Methods allow us to **reuse** the code without retyping the code.

METHOD SYNTAX

Syntax (1)

```
modifier returnType nameOfMethod (Parameter List) {
        // method body
}
```

(2)

```
modifier static returnType nameOfMethod (Parameter List) {
        // method body
}
```

Example

```
public static void main (String[] args) {
        // method body
}
```

METHOD EXAMPLES

```
public class Project{
    public void createProject(){
        System.out.println("New project has been created!");
    }

    public void printAllProjects(){
        System.out.println("College Management System");
        System.out.println("Car Driving Test Game");
        System.out.println("Online Payment System");
    }
}
```

METHOD SYNTAX DESCRIPTION

- **MODIFIER**: Defines access type of the method i.e. from where it can be accessed in your application.
- In Java, there 4 type of the access specifiers.
  - **public**: accessible in all class in your application.
  - **protected**: accessible within the class in which it is defined and in its subclass(es)
  - **private**: accessible only within the class in which it is defined.
  - **default** (declared/defined without using any modifier) : accessible within same class and package.

METHOD SYNTAX DESCRIPTION

```
public void printName() {

}
```

```
protected void printName() {

}
```

```
private void printName() {

}
```

```
void printName() {

}
```

METHOD SYNTAX DESCRIPTION

- **Return type** : The data type of the value returned by the method or void if does not return a value.
- **return** is a reserved keyword in Java i.e, we can't use it as an identifier. It is used to **exit** from a method, with or without a value.
- If the method does not return a value, its return type is void.

RETURN TYPE

```java
public void printName() {
    System.out.println("My name is John");
}
```

```java
public String getName() {
    return "My name is John";
}
```

METHOD SYNTAX DESCRIPTION

- **Method Name** : The name of the method which should be related to the task or operation. E.g add(), delete, play()
- **Parameters** : Comma separated list of the input parameters are defined, preceded with their data type, within the enclosed parenthesis. If there are no parameters, you must use empty parentheses (). It is used to pass the information to method i.e. it is the input the the method.
- **Method body** : it is enclosed between braces. The code you need to be executed to perform your intended operations.

METHOD EXAMPLE

Example: Non return method with parameters

```
public void add(int num1, int num2) {

    int result = num1 + num2;
    System.out.println("Addition Result is: " + result);
}
```

- Here, method name is **add**
- Parameters are num1 and num2 of both int type
- Statements enclosed inside the braces are method body
- This method does not return anything

Now, to call this method:

add(1, 2) or object.add(3, 4)

YOU HAVE TO CALL THE METHOD/FUNCTION

METHOD EXAMPLE

Example: Return type method with parameters

```
public int add(int num1, int num2) {
    int result = num1+num2;
    return result;
}
```

- This method takes two inputs as parameters
- This method returns the result of addition

How to call this method: **int result = add(3, 10);**

METHOD EXAMPLE

Example: Static method without return type

```
public static void myMethod() {
    System.out.println("hello from my method");
}
```

- This method does not return anything
- This method can be called without creating object

How to call this method?

ClassName.myMethod();

METHOD EXAMPLE

Example: Method with parameter without returning

```
public void printMe(String data) {
    System.out.println(data);
}
```

- This method does not return anything
- This method just prints the value of parameter (data)

How to call this method?

printMe(“hello world”);

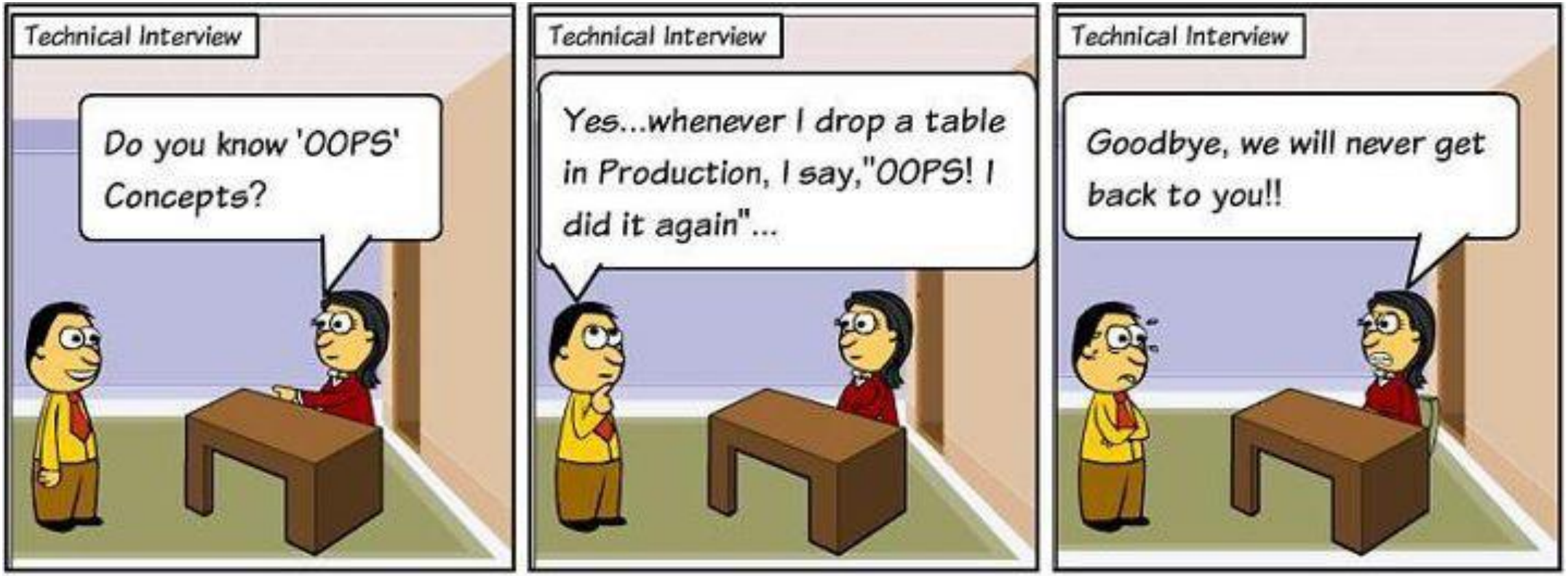
MAIN METHOD

- **public static void main(String[] args)** is the most important Java method.
- When you start learning java programming, this is the first method you encounter.
- Remember the first Java Hello World program you wrote that runs and prints “Hello World”?
- Java main method is the entry point of any java program. Its syntax is always **public static void main(String[] args)**
- A Java program needs to start its execution somewhere. A Java program starts by executing the main method of class.

WHAT WE LEARNED

- Object Oriented Programming (OOPs)
- Class
- Object
- Method

OOPS


Technical Interview
Do you know 'OOPS' Concepts?
Technical Interview
Yes...whenever I drop a table in Production, I say,"OOPS! I did it again"...
Technical Interview
Goodbye, we will never get back to you!!

END OF LECTURE 4

Any questions?