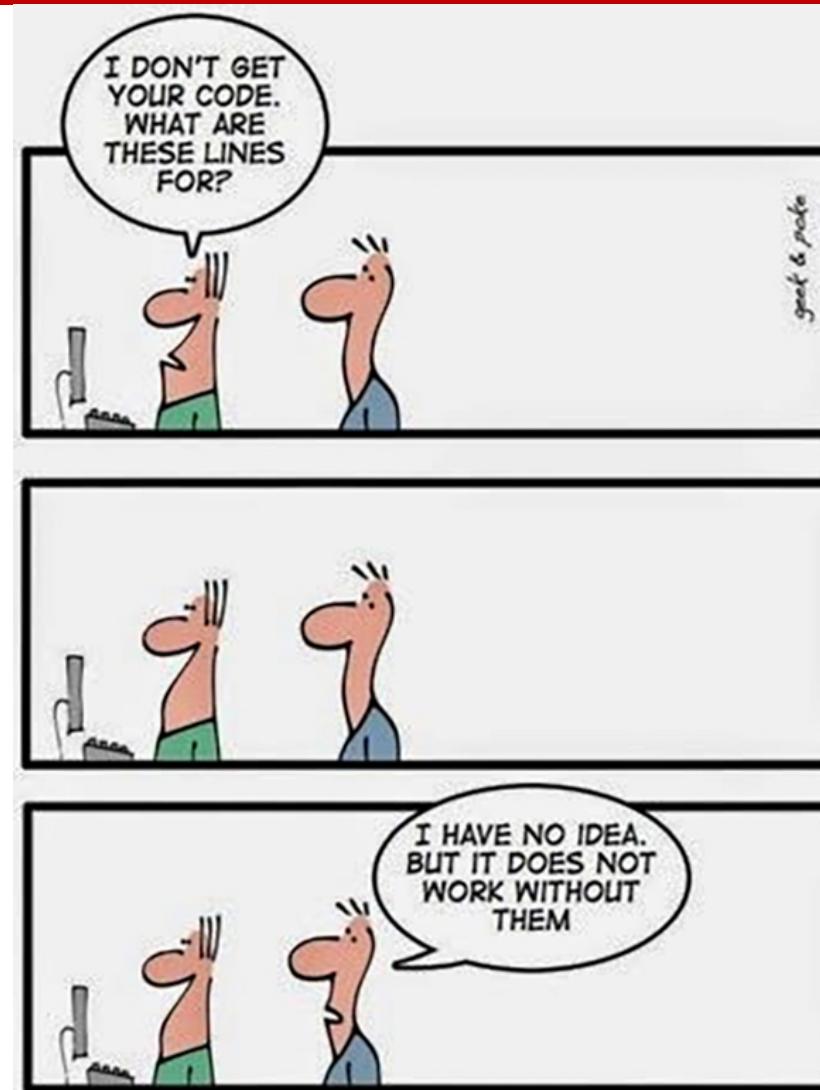


PROGRAMMING

Lecture 3

Sushil Paudel

TILL NOW



PREVIOUS TOPICS

- Separators
- Comments
- Identifiers
- Keywords
- Java Class Library
- Syntax
- Indentation
- Variable
- Data types
- Operators

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, hungry) and behavior (barking, fetching, wagging tail).
- Bicycles also have state (current gear, current pedal cadence, color, model, etc.) and behavior (changing gear, changing pedal cadence, applying brakes).
- 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

- Take a minute right now to observe the real-world objects that are in your area.
- For each object that you see, ask yourself two questions:
 - "What possible states can this object be in?"
 - "What possible behavior can this object perform?"
- Make sure to write down your observations.
- As you do, you'll notice that real-world objects vary in complexity; your desk lamp may have only two possible states (on and off) and two possible behaviors (turn on, turn off), but your radio might have additional states (on, off, current volume, current station) and behavior (turn on, turn off, increase volume, decrease volume, scan, and tune).

OBJECTS



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 - Each bank account has some behaviors, it can be opened, closed, deposited, withdrawn, etc.

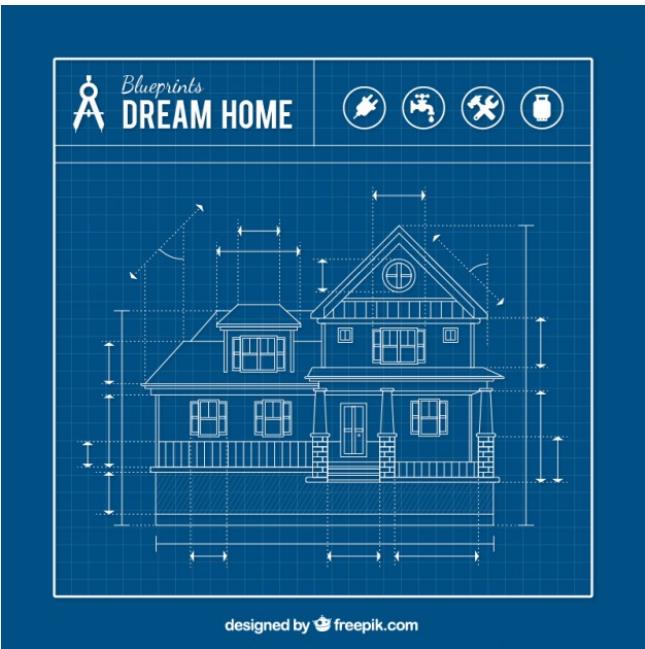
CLASS

- A class is a blueprint, a detailed description, a definition.
- A class describes what an object will be.
- We cannot talk about objects without talking about classes.
- 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:

```
Modifiernname 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**.

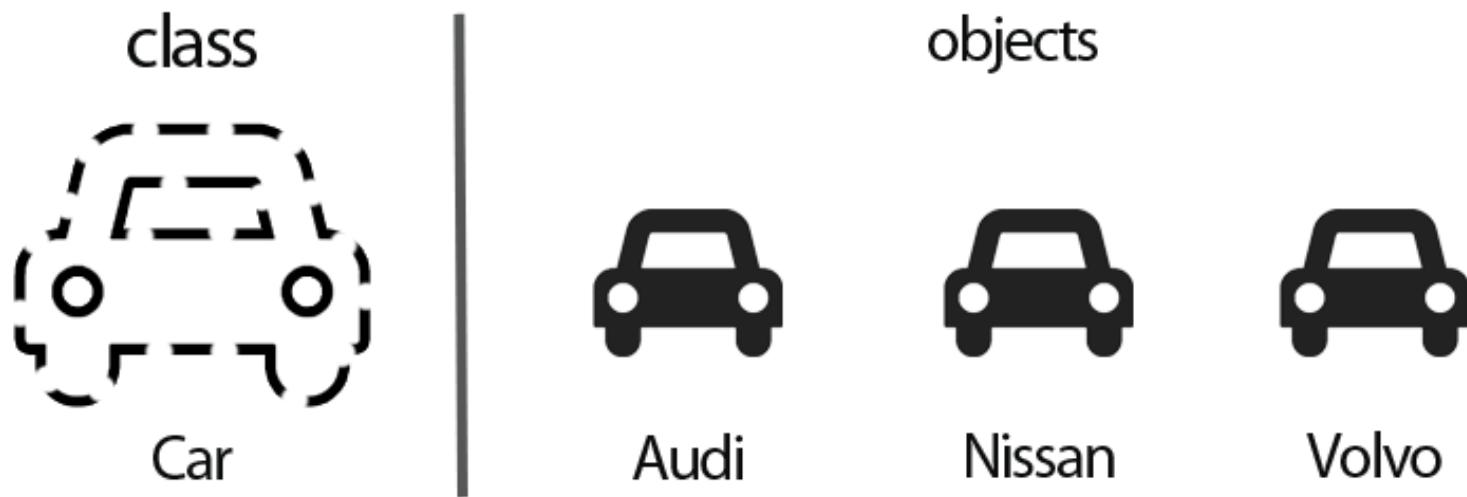
STATE



BEHAVIOR

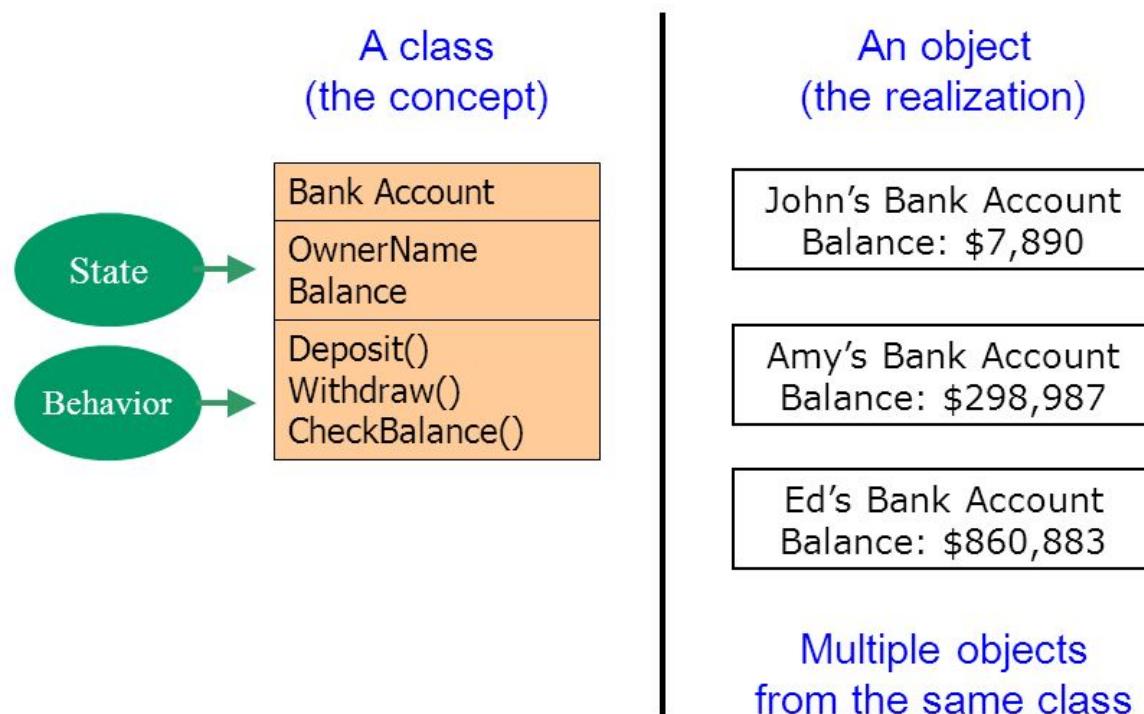


CLASS AND OBJECT

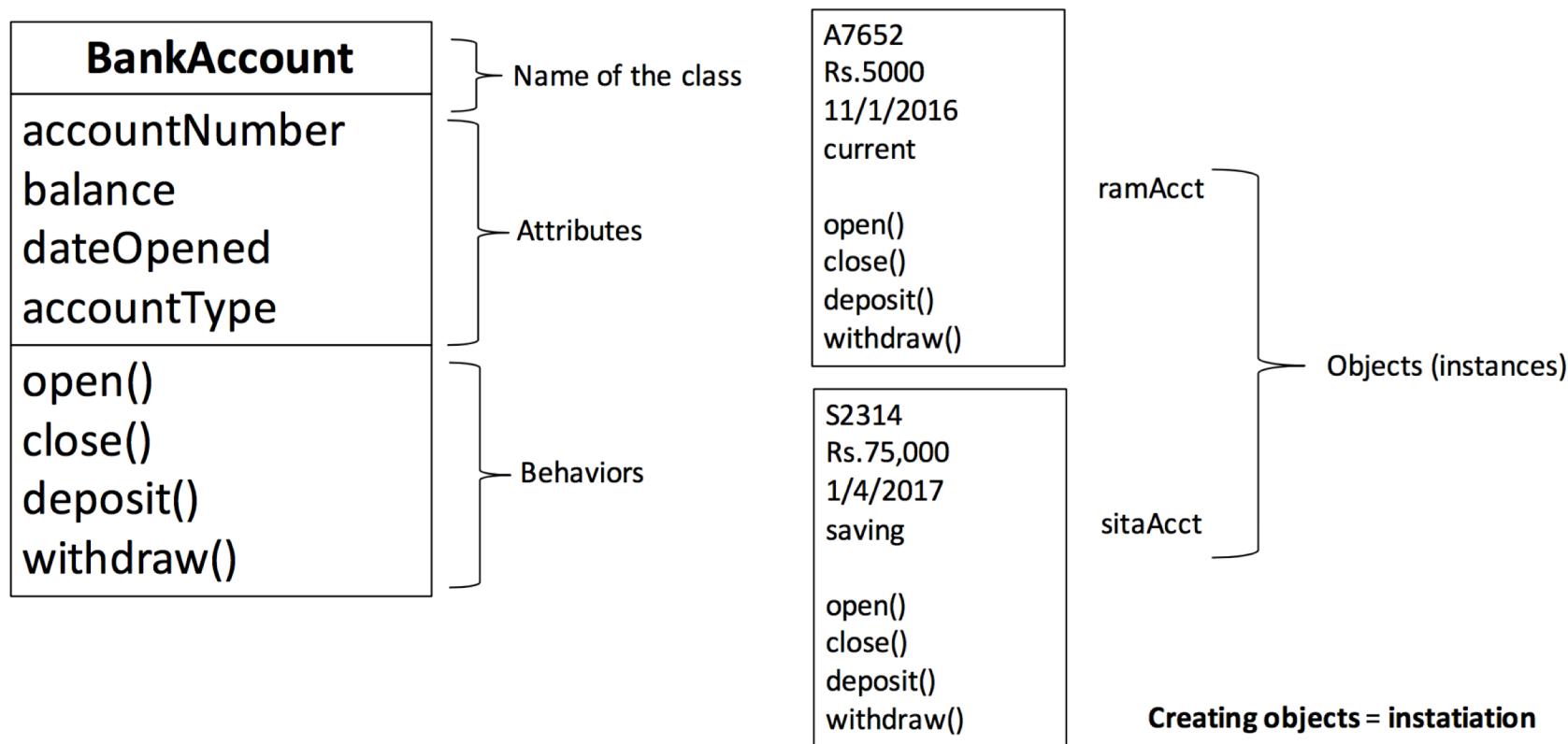


CLASS AND OBJECT

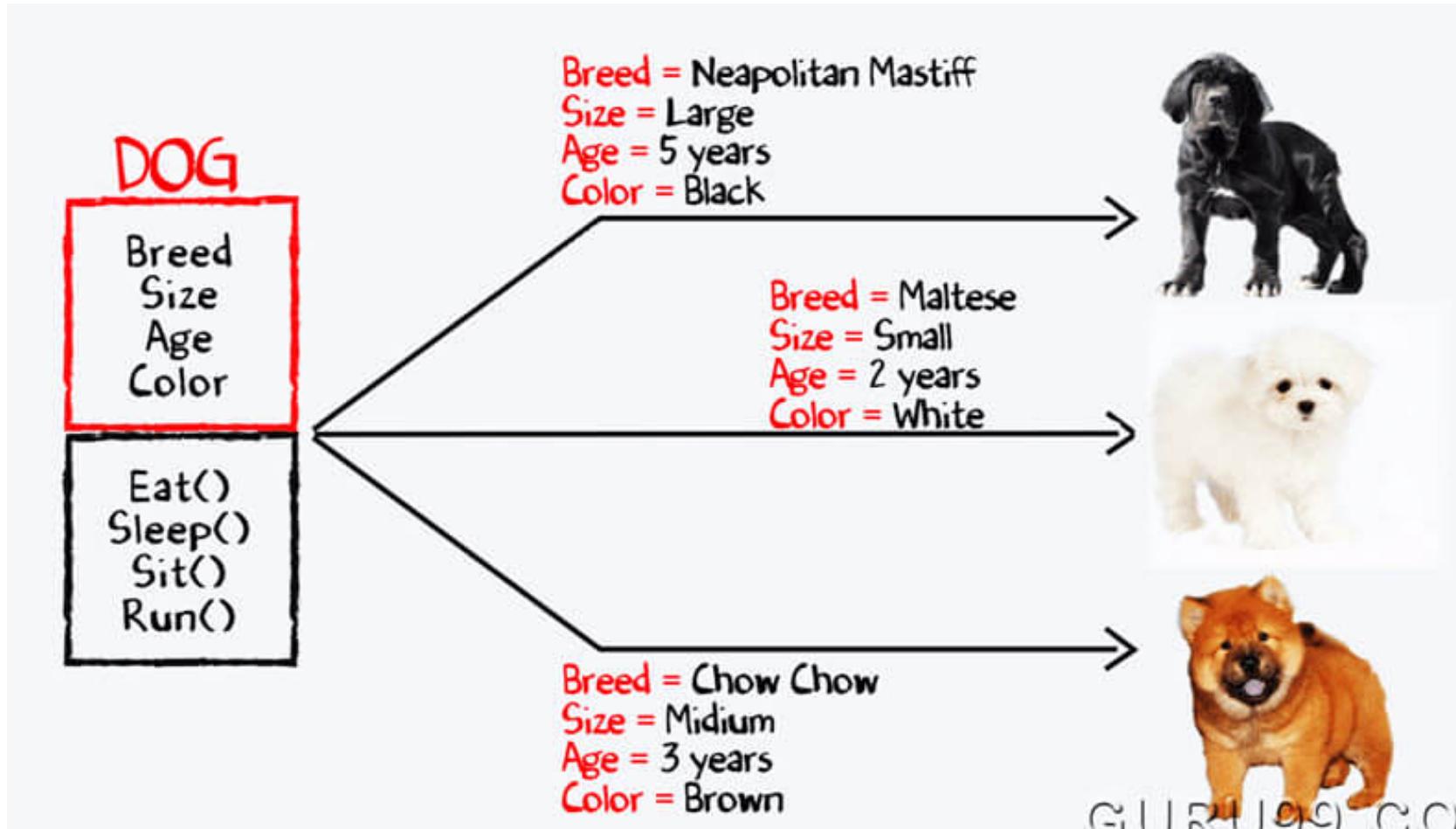
Example



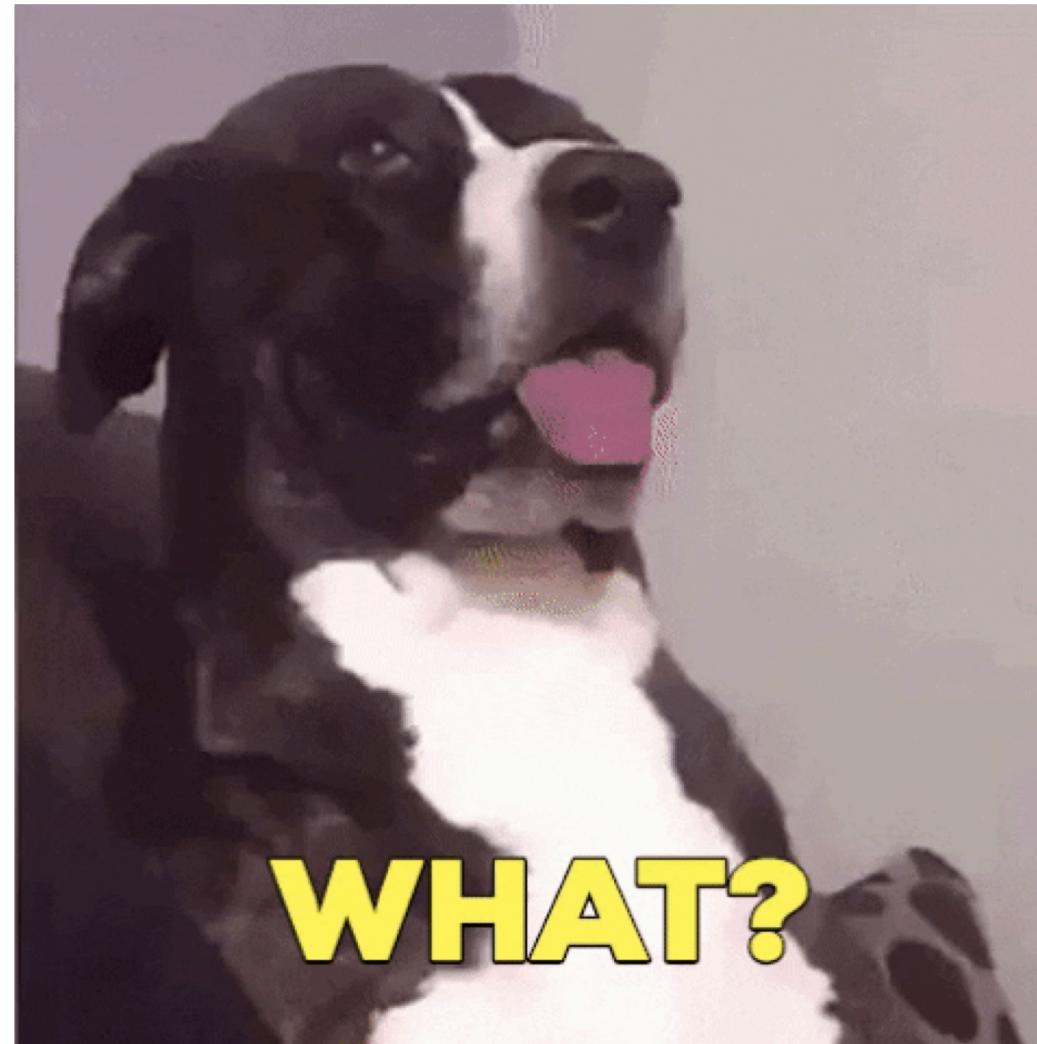
CLASS AND OBJECT



DOG



DOG



CREATE OBJECT

- **Syntax:**

```
ClassName objectName = new ClassName();
```

- **Example:**

```
Animal animal = new Animal();
```

- **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

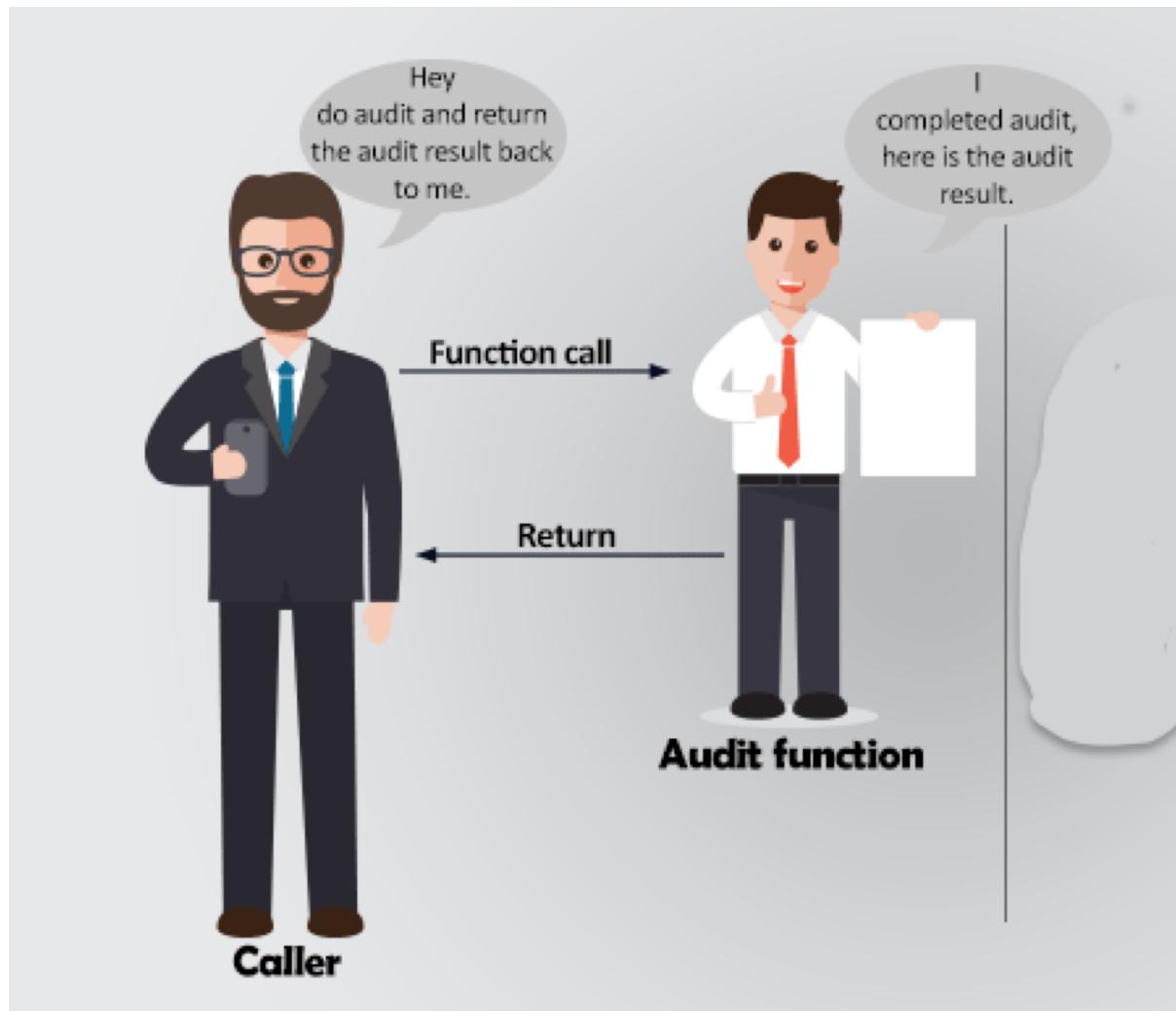
- **Return type** : The data type of the value returned by the method or void if does not return a value.
- **return** is a reserved keyword in Java i.e, we can't use it as an identifier. It is used to **exit** from a method, with or without a value.
- It can return native data types (int, float, double etc.), native objects (String, Map, List etc.), or any other built-in and user defined objects.
- If the method does not return a value, its return type is void.

RETURN TYPE

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

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

RETURN TYPE



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 to 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

**"Why is my
function not
outputting
anything?"**

**"Oh I never called
the 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");`

METHOD EXAMPLE

Example: Return some fixed value

```
public int getTotalPercentage() {  
    return 100;  
}
```

- This method just returns a value i.e. 100

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.

QUESTIONS

- What are the methods/behaviors of MobilePhone class?
- What are the methods/behaviors of Booking (like online booking) class?

WHAT WE LEARNED

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

EXAMPLE

```
// Declaring the class Bank
public class Bank{

    // No return type, just prints the message
    public void openAccount(){
        System.out.println("I am opening the account...");
    }

    // Returns the available balance
    public double getRemainingBalance(){
        return 1123123.12;
    }
}
```

OOPS



END OF LECTURE 3

Any questions?