Lesson:

Java OOPs

PW SKILLS

Pre-Requisites

- Basics of Java
- Functions

Concepts Involved :

- Object and Class
- Passing class to function
- Access Modifiers - Public, Private, Protected, Default
- Getters and Setters
- 'this' keyword
- Constructor
- final keyword
- static keyword
- static functions

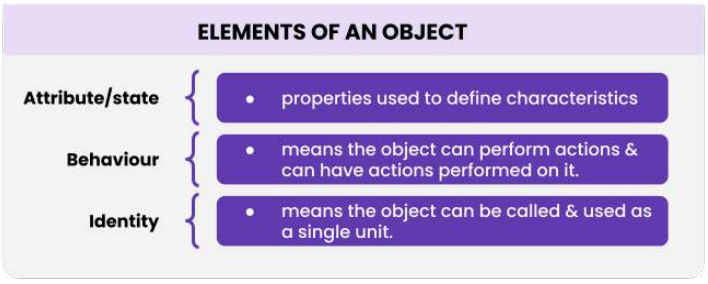
Object and Class

In object-oriented programming, we design a program using objects and classes. Often there is confusion between classes and objects. Let's clear that out.

What is an Object?
An entity that has a state and behavior is known as an object e.g., a chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). Basically Intangible means that we can't touch that thing but it is still there like the 'banking system'. A particular type of bank can be an example of an object but it can't be touched but can be felt by seeing its various behaviors. Don't worry if you have not understood it yet. We are gonna see a lot of examples in this lesson to know the crux of the topic.

Characteristics of an object:

- **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.

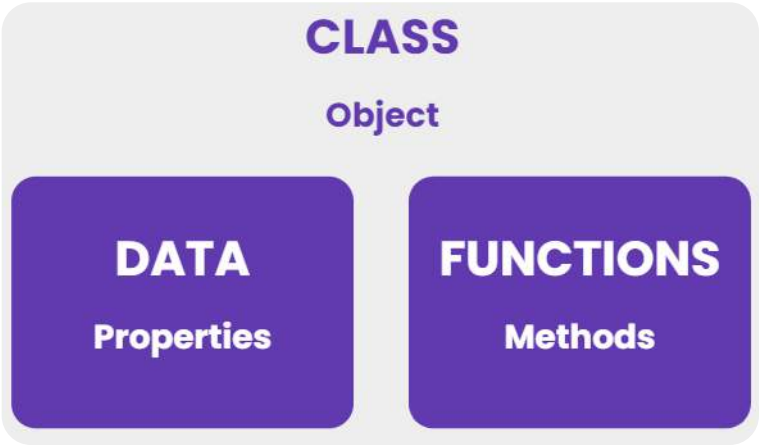


Let's understand the definition of class also and then we will understand both of these terms using practical examples with the help of code.

Class:

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. Using classes, you can create multiple objects with the same behavior instead of writing their code multiple times. This includes classes for objects occurring more than once in your code.



Let's now understand class and object through examples.
-> To create a class we use keyword 'class'

```
public class Main {
    int x = 5;
}
```

The above code will create a class with name as Main and variable x equal to 5.
Now, let's create an object of the above class and relate the code of class and object to understand how they are related.

```
public class Main
{
    int x = 5;
    public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
}
}
```

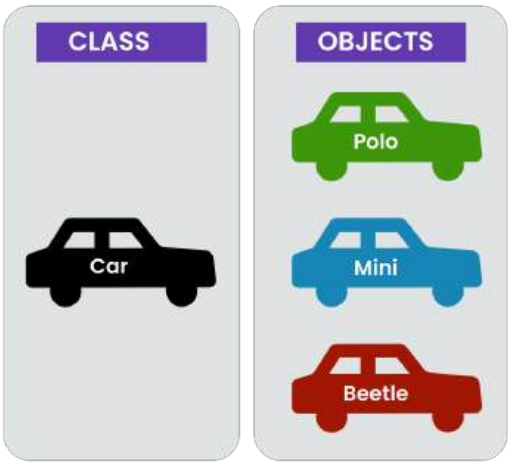
Output:-

```
5
```

The above code shows creation of an object named myObj which is of type Main. if we see above then we will see that the Main class contains a variable x equal to 5. So, we can access the value x by using the '.' operator. myObj.x will be equal to 5 in this case as assigned by us previously. We can change this value by writing any value in front of myObj.x in the main function.

Till now, If you have understood the difference between object and class and how they are related, then fine but if you still have doubt about it, then see below examples.

Example 1
let's see example of a Car. You all know that Car in general is a vague term. Why? Because car can be of many types/models. So, we can call car as a class and different models of car as object. Car is a blueprint on which different models of the car have been made up of. Car doesn't exist physically in general but exist as different models in real life.



Example 2
Similarly, Take another example of Animal, if you will see an animal then after seeing it you will know that which type of animal it is. It can be dog, cat, lion, goat etc. but in general terms it is an animal. So, Animal can be called as a Class and different types of animals like cat, dog etc. will be termed as objects. Different objects have different behaviors. Dog barks. Lion roars. They have different colors.

Example of Animal class

Output:

```
Dog Barks
Cat Meows
Lion Roars
```

A code is given below, try to understand in on your own first. After seeing the code, see the explanation given below to understand the main concept behind this code.

JAVA LP48 CODE3

Output:

```
led on? true
halogen on? false
```

Explanation of the code: In this code, we have made a class named Lamp. It has two functions named turnOn and turnOff. Using these functions we can on or off the Lamp. Now we have made two objects of type Lamp. One object is named as led and other is halogen. Now we have applied those functions on the objects and printed the output.

Naming Convention for Class and Object

For Class:

- Name of the class should start with an uppercase letter.
- Use upper camel casing in case the name of the class contains more than one word. e.g BankController.
- The class name should give us an idea of the code in the body of class.
- The name of the class should be a noun.
- e.g Student, Bank

For Object:

- Name of the object should start with a lowercase letter.
- e.g For class Student, we can give object names like student1,student2 etc. in a general way.

Ways to create a Class in Java

1.) Main within the class

Main function is created inside the class. This type of creation of class is less used as the class created cannot be used again as it also contains the main function.

JAVA LP48 CODE4

2.) Main outside the class

Main function is created outside the class.In real time development, we create classes and use the object of these classes in another class. It is a better approach than the previous one because this has the advantage that we can reuse our .class file somewhere in other projects without compiling the code again

JAVA LP48 CODE5

A class in java can contain:

• data member
• method
• constructor
• nested class

Ways to create an Object in Java

There are four ways to create objects in java. Basically an object can be created by only new keyword and the rest of the ways to create an object internally use new keyword.

- Using new keyword: It is the most common and general way to create an object in java.
  JAVA LP48 CODE6

- **Using Class.forName(String className) method:** There is a pre-defined class in java.lang package with name Class. The forName(String className) method returns the Class object associated with the class with the given string name. We have to give a fully qualified name for a class. On calling newInstance() method on this Class object returns a new instance of the class with the given string name.
  JAVA LP48 CODE7

- **Using clone() method:** clone() method is present in Object class. It creates and returns a copy of the object.
  JAVA LP48 CODE8

- **Deserialization:** It is a technique of reading an object from the saved state in a file.
  JAVA LP48 CODE9

Passing class to function

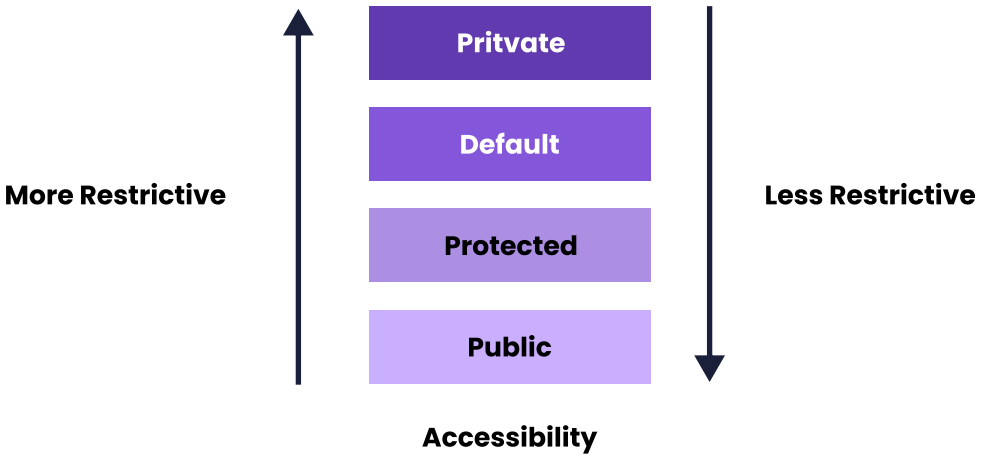
In order to pass the class to a function following conditions must be met:

1. Declare the class outside the main.
2. Classes are passed by reference.

Access Modifiers - Public, Private, Default

In Java, access modifiers are keywords that determine the visibility and accessibility of a class, field, method, or constructor. There are four types of access modifiers in Java:

1. private: A class, field, method, or constructor that is declared as private can only be accessed within the class in which it is declared.
2. protected: A class, field, method, or constructor that is declared as protected can be accessed within the class in which it is declared, as well as by any subclass of that class.
3. public: A class, field, method, or constructor that is declared as public can be accessed from anywhere in the code.
4. default (also known as package-private): A class, field, method, or constructor that does not have an explicit access modifier specified is considered to have default (package-private) access. This means that it can only be accessed within the package in which it is declared.



Let us see each access modifier in detail one by one with a java program.

1) 'private' Access Modifier
The private access modifier in Java is the most restrictive access level. A class, field, method, or constructor that is declared as private can only be accessed within the class in which it is declared.

Here is an example of how the private access modifier can be used in a Java program:

JAVA LP62 CODE2

Output:

```
This is a public method
This is a private method
```

In this code, the MyClass class has a private field called privateField and a private method called privateMethod. The MyClass class also has a public method called publicMethod, which calls the privateMethod. The Main class has a main method that creates an instance of MyClass and calls the publicMethod. Because the publicMethod is public, it can be accessed from any other class in the code. However, the privateMethod is private and can only be accessed within the MyClass class. As a result, attempting to call privateMethod directly from the Main class will cause a compile-time error.

2) 'protected' Access Modifier
The protected access modifier in Java is similar to the private access modifier, with the added ability to be accessed by subclasses of the class in which it is declared. A class, field, method, or constructor that is declared as protected can be accessed within the class in which it is declared, as well as by any subclass of that class.

Here is an example of a class with a protected field and a protected method:

JAVA LP62 CODE3

Output:

```
This is a public method
This is a protected method
```

In this code, the MyClass class has a protected field called protectedField and a protected method called protectedMethod. The MySubClass class is a subclass of MyClass and has a public method called publicMethod, which calls the protectedMethod. The Main class has a main method that creates an instance of MySubClass and calls the publicMethod. Because the publicMethod is public, it can be accessed from any other class in the code. However, the protectedMethod is protected and can only be accessed within the MyClass class and its subclasses. As a result, attempting to call protectedMethod directly from the Main class will cause a compile-time error.

The protected access modifier is often used to allow subclasses to access certain members of a class, while still preventing access from other classes. This can be useful for implementing inheritance and building a class hierarchy.

3) 'public' Access Modifier
The public access modifier in Java is the most permissive access level. A class, field, method, or constructor that is declared as public can be accessed from anywhere in the code, regardless of whether it is in the same package or a different package.

Here is an example of a class with a public field and a public method:

JAVA LP62 CODE1

Output:

```
This is a public method
```

In this code, the MyClass class has a public field called publicField and a public method called publicMethod. The Main class has a main method that creates an instance of MyClass and accesses the publicField and publicMethod. Because both the field and the method are public, they can be accessed from any other class in the code.

It is important to note that, while the public access modifier allows for the greatest level of accessibility, it should be used with caution. Overuse of the public access modifier can lead to poor encapsulation and maintainability of the code. It is generally a good practice to use the private access modifier for fields and methods that do not need to be accessed from outside the class, and to use public access only when necessary.

4) 'default' Access Modifier
The default (also known as package-private) access modifier in Java is the access level that is applied when no explicit access modifier is specified. A class, field, method, or constructor that does not have an explicit access modifier specified is considered to have default (package-private) access. This means that it can only be accessed within the package in which it is declared.

Here is an example of a class with a default field and a default method:

JAVA LP62 CODE4

Output:

```
This is a default method
```

In this code, the MyClass class has a default field called defaultField and a default method called defaultMethod. The Main class is in the same package as MyClass and has a main method that creates an instance of MyClass and accesses the defaultField and defaultMethod. Because both the field and the method have default access, they can be accessed from any other class in the same package.

The default access modifier is often used to limit the visibility of a class, field, method, or constructor to only those classes in the same package. This can be useful for organizing code into logical packages and controlling the accessibility of certain members.

Getters and Setters

Getter and Setter are methods used to protect your data and make your code more secure. Getter returns the value (accessors), it returns the value of data type int, String, double, float, etc. For the program's convenience, getter starts with the word "get" followed by the variable name.

While Setter sets or updates the value (mutators). It sets the value for any variable used in a class's programs. and starts with the word "set" followed by the variable name. Getter and Setter make the programmer convenient in setting and getting the value for a particular data type. In both getter and setter, the first letter of the variable should be capital.

Getter and Setter give you the convenience of entering the value of the variables of any data type by the requirement of the code. Getters and setters let you manage how crucial variables in your code are accessed and altered.

```
public class Vehicle {
  private String color;

  // Getter
  public String getColor() {
    return color;
  }

  // Setter
  public void setColor(String c) {
    this.color = c;
  }
}
```

The getter method returns the value of the attribute. The setter method takes a parameter and assigns it to the attribute.

'this' keyword

In Java there are a lot of keywords and some of them are really important like 'static' and 'this'. Let us study about them in detail.

What is a keyword ?

Keyword is a word which is reserved by the programming language. That is, a keyword has a special meaning so it can't be used as the name of some variable/class/function. A keyword has a special meaning which defines specific parameters of the program.

Keywords in Java

In the Java language, We have a lot of keywords such as static,this,public, long,protected,boolean etc. These keywords are used to define access of objects/classes/functions or to define some important values in the

program e.g boolean keyword can take only two values whether true or false. So, it can be used in code where we require to assign these values to a variable or where we require return type of function to be either true or false. We will discuss “static” and “this” keyword in detail in this lecture.

List of Java Keywords

boolean	byte	char	double	float
short	void	int	long	while
for	do	switch	break	continue
case	default	if	else	try
catch	finally	class	abstract	extends
final	import	new	instance of	private
interface	native	public	package	implements
protected	return	static	super	synchronized
this	throw	throws	transient	volatile

What is the ‘this’ keyword?

‘this’ keyword refers to the current object in a method or constructor.
The most common use of this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).
Don’t understand ? No problem. Think in this way. If you are creating an object of a class and now you want to change the attributes of the class then what will you do ? Will you change class attributes or only object attributes ? Obviously, you will only change object attributes. So, this keyword denotes the object's attributes instead of class attributes while writing code.
For example, if a car has different models, then using this keyword you can assign a model its specific attributes like model_name, color, price etc. which will be specific to that model.
So, when we initialize a parameterized constructor, then we will take care that this keyword is used because it makes it easy to see the code and to understand what is actually happening in the code.

Uses of ‘this’ keyword

The ‘this’ keyword can be used for following :

1.Using ‘this’ keyword to refer current class instance variables
JAVA LP50 CODE

The code shows how this keyword is used to initialize an object’s attributes.
Output :

```
a = 15   b = 28
```

2.Using this() to invoke current class constructor
JAVA LP50 CODE7

In this code, this keyword is used from a non parameterized constructor to a parameterized constructor.

Output :

```
Inside parameterized constructor
Inside default constructor
```

3.Using 'this' keyword to return the current class instance
JAVA LP50 CODE8
The above code shows how current class instance is printe with the help of this keyword.

Output:

```
a = 15   b = 28
```

4.Using 'this' keyword as method parameter
JAVA LP50 CODE9
The code shows how this keyword can be used as a parameter in the constructor.

Output :

```
a = 15   b = 28
```

5.Using 'this' keyword to invoke current class method
JAVA LP50 CODE10
The above code shows how 'this' keyword can be used invoke a current class method.

Output:

```
Inside show function
Inside display function
```

6. Using 'this' keyword as an argument in the constructor call
JAVA LP50 CODE11
Using this keyword as an argument in the constructor call. A constructor of A has been called without creating of class.

Output:

```
Value of x in Class B is given as : 51
```

Constructor

As known from the word constructor, it means to build something. Similarly, In Java, constructor is used to initialize the object. Seems confusing? No problem. Let's see below how a constructor initializes an object and first of all why we need a constructor. We will cover every detail about Java constructor here. So, Let's start with the basics!!

What is a Constructor?

A constructor in Java is defined as a method which is used to initialize objects. A constructor is called when an object of the class is created. Constructor can be used to set values initially for attributes of object.

Whenever we create a new object, at least one constructor is called. Confused why at least one constructor is called everytime we create an object? Because if we will not specify any constructor in the class then Java calls a default constructor. That is the reason for calling of a minimum of one constructor while creating an object.

Need of a Constructor

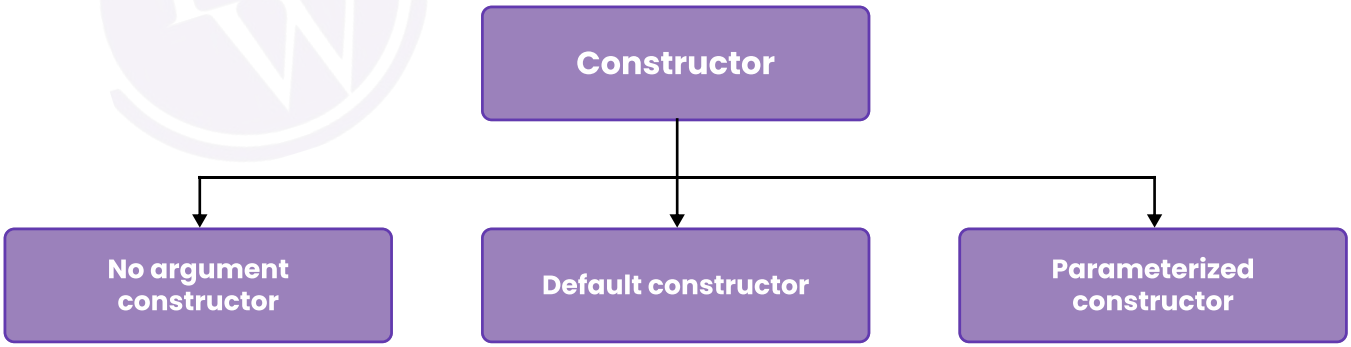
As you know, class is a blueprint using which object is created. Class does not exist in real life but objects do exist. So, Let's assume we have created a class named "Room". So, this class will have attributes like roomLength, roomWidth and roomHeight. Now, when we will create an object of the class "Room" then as we know that the object will be present in real in computer memory , so attributes like roomLength, roomHeight and roomWidth can be initialized with some value.Now, the object can only be created if the constructor will be present in the class because constructor is called first when we are creating object and is necessary for object creation. This is where we need a constructor. Constructor will be called whenever an object is created and attributes of the object will be initialized with given values in the constructor. So constructors are used to assign values to the class variables at the time of object creation, either explicitly done by the programmer or by Java itself (default constructor).
We will see how a default constructor is made and different types of constructors in next parts. Keep reading.

Rules to follow while creating a constructor

- A constructor in Java must be named the same as its class name.
- A constructor must not have an explicit return type.
- A Constructor cannot be abstract, static, final and synchronized.
- Access modifiers can be used while creating a constructor.

Types of constructor

1.) No argument constructor
2.) Parameterized constructor
3.) Default constructor



1) No-argument constructor

A constructor that has no parameter is known as the No-argument or Zero argument constructor. If we don't define a constructor in a class, then the compiler creates a constructor(with no arguments) for the class. And if we write a constructor with arguments or no arguments then the compiler does not create a default constructor.

Code: JAVA LP49 CODE1

Output

```
Constructor called
null
0
```

Explanation :
In the above code, we have made a constructor with no argument. So, whenever an object is created of the class 'Bank' then "Constructor called" will be printed. As we have not assigned any initial value to the attributes of the Bank class, therefore id = 0 and name = null have been assigned by java on its own which are the default values for the corresponding data types. So, that is how initialization works if we do not assign any value to the attributes.

2) Parameterized constructor
A constructor that has parameters present in its representation is known as a parameterized constructor. If we want to initialize the fields of the class with our own values, then we use a parameterized constructor.

Code: JAVA LP49 CODE2

Output:

```
SBI
1104
```

Explanation:
In this code, we have used a parameterized constructor. How ? We have the constructor that is taking two arguments, and it will assign those values to the corresponding attributes of the object while initializing it. So, we can initialize the object's name and id according to our use case.

3) All Argument Constructor
This is a special type of parameterized constructor which takes all attributes of a class as parameters.All Argument constructor is a constructor with 1 parameter for each field in your class i.e whichever attributes we have written in the class will be used as arguments in All Argument Constructor.

4) Default Constructor
A default constructor is invisible. A default constructor is a no argument constructor, but is not present explicitly in the class. It is made by java if no constructor will be written by us. So, if we write any constructor for initialization, then there will be no default constructor.

Code: JAVA LP49 CODE3

Explanation:
In this code, we have not defined any constructor. So, when we are making an object, then a default constructor has been called out which will assign default values to the class attributes corresponding to their data types.

Constructor Overloading

In Java, constructor overloading is a technique of having more than one constructor with different parameters . They are arranged in a way so that each constructor performs a different task. These are differentiated by the compiler by the number of parameters taken and their types.

Data()

Data(int da)

Data(String na)

- Overloaded constructors have the same name(name of the class) but different number (or types or both) of arguments.
- Depending upon the number and type of arguments passed, specific constructor is called.

Code: [JAVA LP49 CODE4]

Output:

```
Constructor with one argument : String : Rahil
Constructor with two arguments :  String and Integer : Rahil 45
Constructor with one argument : Long : 100000000
```

Explanation:
In this code, we have created three constructors. Why? because when we are calling the object, then there can be only one argument given, or there can be an argument with a different type, or there can be more number of arguments. So, Accordingly we have written the constructor for the class.
So, we have done constructor overloading here. That is , a constructor will be called according to the parameters given while initializing the object. The decision of which constructor to call is taken at runtime.

Can Constructor be 'static' in java?
No, a constructor in Java cannot be declared as static.

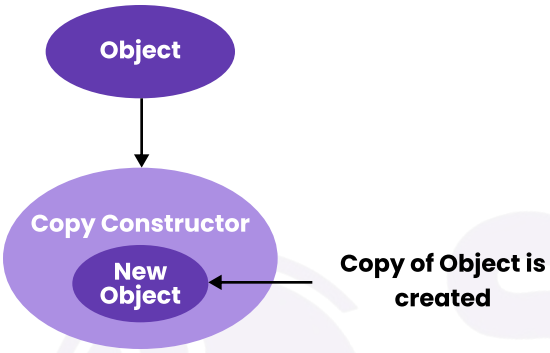
Why?
A constructor cannot be declared as static because it is associated with the creation of an instance of an object, and a static method is associated with a class rather than an object. A static method can be called without creating an instance of the class, but a constructor can only be called when an instance of the class is created.

Differences between Constructors and Methods

The definitions of constructors and methods look similar in code. They can take parameters and they have bodies in braces. But, despite appearing similar, here are the differences between them :

Constructors	Methods
1. A constructor is used to initialize the object. 2. A constructor must not have a return type. 3. A constructor is invoked without calling it explicitly. 4. Constructor must have the same name as the class name. 5. Constructor can be initialized by default if not initialized.	1. A method is used to show some behavior of the object. 2. A method must have a return type. 3. A method is invoked if we call it explicitly. 4. A method can have different names. 5. A method should be written, and will not be initialized by default.

What is Copy Constructor?
It basically means that we can copy values of the object created before in the new object only by specifying that object in argument. Let's understand this through code.



Code: JAVA LP49 CODE5

Output:

```
Copy constructor has been called
(45.0 + 91.0i)
```

Explanation:
Here, we have created a class ComplexNumber, Our main objective is to copy the values of object c1 to object c2 without using a parameterized constructor. So, what we are doing is we are creating an object c1 and then we are copying the values of attributes into a newly created object named c2.

What is Destructor?
A destructor is a special type of function or method in some programming languages(C,C++), that is used to clean up and release resources when an object is no longer needed. Destructors are often used to deallocate memory, close file handles, and release other resources that were acquired during the lifetime of an object.

In Java, Destructor is not present because it has an automatic garbage collector which destroys resources which will no longer be needed.

Final keyword

The final keyword is a non-access modifier used for classes, attributes and methods, which makes them non-changeable (impossible to inherit or override).

The final keyword is useful when you want a variable to always store the same value, like PI (3.14159...).

The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

1. variable
2. method
3. class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only.

Characteristics of final keyword in java:
In Java, the final keyword is used to indicate that a variable, method, or class cannot be modified or extended. Here are some of its characteristics:

- **Final variables:** When a variable is declared as final, its value cannot be changed once it has been initialized. This is useful for declaring constants or other values that should not be modified.
- **Final methods:** When a method is declared as final, it cannot be overridden by a subclass. This is useful for methods that are part of a class's public API and should not be modified by subclasses.
- **Final classes:** When a class is declared as final, it cannot be extended by a subclass. This is useful for classes that are intended to be used as is and should not be modified or extended.
- **Initialization:** Final variables must be initialized either at the time of declaration or in the constructor of the class. This ensures that the value of the variable is set and cannot be changed.
- **Performance:** The use of final can sometimes improve performance, as the compiler can optimize the code more effectively when it knows that a variable or method cannot be changed.
- **Security:** final can help improve security by preventing malicious code from modifying sensitive data or behavior.

Overall, the final keyword is a useful tool for improving code quality and ensuring that certain aspects of a program cannot be modified or extended. By declaring variables, methods, and classes as final, developers can write more secure, robust, and maintainable code.

Static keyword

What is 'static' keyword?
The static keyword is mainly used for memory management in Java. A static keyword can be applied with variables, blocks, functions and class. The static keyword is a property of a class rather than an instance of the class. The static keyword is used for a constant variable or a method that is the same for every instance of a class.
We will study about it in detail with examples to understand it deeply.

JAVA LP50 Code1

The code shows how we are calling the function without creating an object of it.

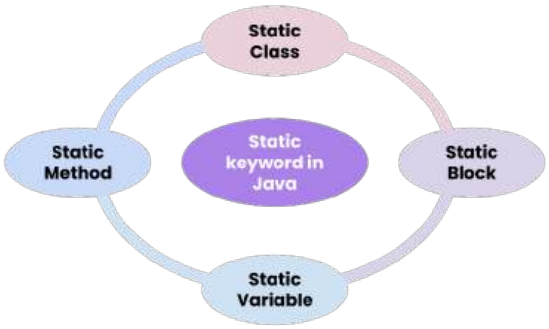
Output :

```
from check
```

Where is the 'static' keyword applicable?

The static keyword is a non access modifier in java which is applicable for the following :

1. Variables
2. Functions
3. Blocks
4. Class



Static Variables

If we declare any variable as static, then it is called a static variable. When a variable is declared as static, then a single copy of that variable is created and shared among all of the objects at the class level. static variables are global variables. All instances of the class share the same static variable.
We can create static variables at the class level only. Static blocks and static variables are executed according to the sequence in which they are written in the program.

Why static ? It makes our program more efficient as every object doesn't allocate separate memory to a static variable.

Let's understand with an example. Assume there is a school which has 1000 students. Now every student is a separate object. Each student has attributes like name, ID and principal name. Now, let's assume that the principal of that school is changed. So, As every student in a school has the same principal, every student's principal will be updated to the new principal name. But, this will consume a lot of memory and computation. So, what we do is, we will make the principal name as static which means that it is a property of the class. So, when one student's principal has changed, then it will automatically change the principal name for all students without doing it separately for each of them. By doing this, it will save memory and computation.
JAVA LP50 CODE2

The code shows that the static block is executed before the main function.

Output:

```
from n
Inside static block will be given priority
Value of a : 100
from main
```

Static functions

When a method is declared with the static keyword, it is known as the static method. The most common example of a static method is the main( ) method. As We know from our discussion above, Any static member can be accessed before any objects of its class are created, and without referencing any object. Functions declared as static have some properties like:

- They can only directly call other static functions.
- They can only directly access static data.
- They cannot refer to this or super in any way.
- A static method belongs to class rather than object.
- A static method can access static data member and can change its value.

JAVA LP50 CODE3

The code shows how changing college name changes the name of college for all students showing static is a property of class.

Output:

```
145 Karmanya IIIT
235 Aryan IIIT
376 Rajkumar IIIT
145 Karmanya NSUT
235 Aryan NSUT
376 Rajkumar NSUT
```

Static Block

It is used to initialize static data member. It is used to initialize before the main method at the time of loading of class. It gets executed only once when class gets loaded. After executing once, it is not needed to be executed again while creating different objects.

JAVA LP50 CODE4

The above code shows that the static block is executed first no matter whether it is written below the main function or above.

Output:

```
50 100
```

Static class

We can declare a class static by using the static keyword. A class can be declared static only if it is a nested class. It does not require any reference of the outer class. The property of the static class is that it does not allow us to access the non-static members of the outer class.

To understand the concept of static class well, we have to first understand the meaning of outer class, inner class and nested class. After that we will see how static class can be useful to us while writing java program.

Outer class : The class in which nested class is defined is called outer class.

Inner class : The classes that are non-static and nested are called inner classes. We cannot create an instance of the inner class without creating an instance of the outer class. Without using the reference to the outer class instance, an instance of the inner class can access the members of its outer class. It makes the program simple and concise.

Nested class: A class within a class is known as a nested class. It may be static or non-static. The major difference between static and non-static class is following:

- An instance of the static nested class can be created without creating an instance of its outer class.
- The static and non-static members of an outer class can be accessed by an inner class.
- Static members of an outer class can be accessed by both the outer class and any inner classes, including static inner classes.

JAVA LP50 CODE5

The code shows nested static class. Str variable is called by display function in the nested class.

Output :

```
hello friends!!
```