Java (OOP)

Name: Teresa Jency Bala

ID: 19 385 20 113

Course Code: CSE 1221

Department: CSE

Campus: Imperial College of Engineering, Khulna

Affiliated by Rajshahi University (385)

Date: 29-10-19

Day: Tuesday.

1. What is object?

Any entity that has a state and behavior is known as an object. For example: a pen, a table etc. It can be physical or logical.

An Object is defined as an instance of a class. An object contains an address and takes up some space in memory.

Objects can communicate, without knowing the details of each other's data or code. The only necessary thing is the type of response returned by the objects.

Example: A dog is an object because it has states like color, name, breed etc. As well as as behaviours like wagging tail, barking, eating etc. Pen is an object. Its name is Raynolds; color is white, known as its state. It is used to write, so writing is its behaviour.

Object Definitions:

- An object is a real-world entity
- A Runtime entity
- An entity which has state and behaviour.

An Object has three Characteristics:

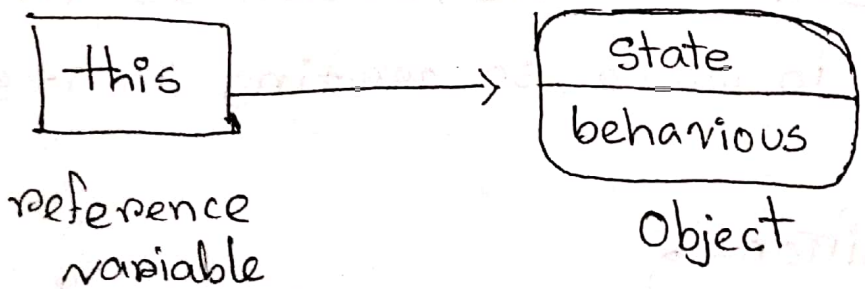
⊗ State: Represents the data (value) of an object

⊗ Behaviour: Represents the behaviour (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.

2. What is this keyword in java?

The this keyword is a reference variable that refers to the current object. There are various use of this keyword in java.



3. What are the main uses of this keyword?

The following are the uses-

- this can be used to refer to the current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke current class constructor.
- this can be passed as an argument in the constructor call.
- this can be used to return the current class instance from the method.

4. What is OOP?

Object Oriented Programming combine data and action together. It works on the principle that objects are the most important part of the program. It allows users create the objects that they want and then create methods to handle those objects. Manipulating these objects to get results is the goal

of OOP

5. What are the 5 design principles from an object-oriented approach from SOLID?

~~Solid~~ SOLID stands for -

S - Single Responsibility Principle

O - Open closed design principle.

L - Liskov. Substitution Principle

I - Interface segregation principle

D - Dependency inversion principle.

6. What are the main OOP concepts in Java?

These are -

① Class

② Object

③ Encapsulation

④ Inheritance

⑤ Polymorphism

⑥ Abstraction

⑦ Package

⑧ Interface.

7. Explain the Basic Concepts in one line.

- Abstraction: The process of picking out (abstracting) common features of objects and procedures.
- Class: A category of objects. The class defines all the common properties of the different objects that belong to it.
- Encapsulation: The process of combining element to create a new entity. A procedure is a type of encapsulation because it combines a series of computer instructions.
- Inheritance: A feature that represents the "is a" relationship between different classes.
- Interface: The languages and codes that the applications use to communicate with each other and with the hardware.
- Polymorphism: An ability to process objects differently depending on their data type or class.
- Object: A self-contained entity that consists of both data and procedures to manipulate the data.

- **Procedure:** A section of a program that performs a specific task

8. What are the common terminology used in OOPs.

- **Class variables:-** Belong to the class as a whole; there is only one copy of each one.
- **Instance variables or attributes:-** Data that belong to individual objects; every object has it's own copy of each one.
- **Members variable:-** Refers to both the class and instance variables that are defined by a particular class.
- **Class methods:-** Belong to the class as a whole and have access only to class variable and inputs from the procedure call.
- **Instance method:-** Belong to individual objects, and have access to instance variable for the specific object they are called on, inputs, and class variables.

9. Point out the characters of Package.

→ A package lets you group classes in subdirectories to avoid accidental name conflicts.

⇒ The package statement must be placed at the first line in the ~~fine~~ code

⇒ If a package statement is omitted from file then the code is part of the default package that has no name.

10. What are the advantages of OOP ?

The advantages of Object Oriented programming are given below-

① OOP provides a clear modular structure for programs.

② It is good for defining abstract data types.

③ Implementation details are hidden from other modules and other modules has a clearly defined interface.

④ It is easy to maintain and modify existing code as new objects can be created with small differences to existing ones.

(5) Objects, methods, instances, message passing inheritance are some important properties provided by these particular languages.

(6) Encapsulation, polymorphism, abstraction are also counts in these fundametals of programming language.

(7) It implements real life scenario

(8) In OOP, programmers not only defines data types but also deals with operations applied for data structures.

(9) Easy to make minor changes in the data representation on the procedures in an object oriented program

(10) Can easily be extended i.e extensible programming. New features or change can be done easily.

11. What is Encapsulation?

The mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

12. What is Abstraction?

Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

13. What is Inheritance?

The process defined ~~as~~ where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

- The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).
- Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of existing class.

The keyword used for inheritance is **extends**.

14. What is polymorphism?

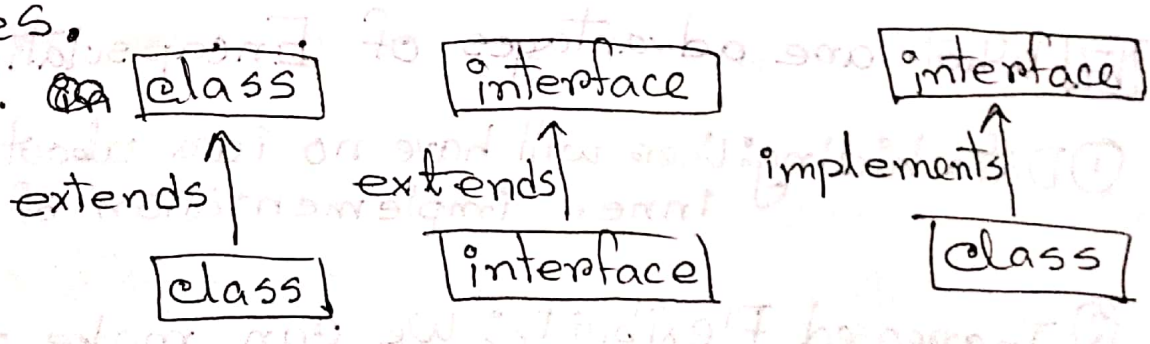
Polymorphism refers to the ability of a variable object or function to take on multiple forms.

In Java, we use method overloading and

method overriding to achieve polymorphism. For example when we comand to speak a cat speaks "meow", dow barks "woof", duck speak "quack" etc.

15. What is Interface?

An Interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface. Along with abstract methods, as interface may also contain constants, default methods, static methods, and nested types.



16. How interface is both similar & dissimilar to class?

Similarities betn Interface & Class:

1. Interface can have any number of method.
2. It has same file extension as class (.java)

Dissimilarities betn Interface & Class:

1. Interface cannot be instantiated. (object of interface cannot be created)
2. Interface doesn't contain constructor
3. All methods in interface are abstract
4. An interface can't have instance variables.
5. An interface can extend multiple interface.

17. What are advantages of Encapsulation?

① Data hiding: User will have no idea about the inner implementation of class.

② Increased Flexibility: We can make variables and methods read-only or write-only as per requirment.

③ Resuability: Easy to reuse and easy to change with new requirement.

④ Testing is easy.

18. Difference between Abstraction & Encapsulation.

Abstraction		Encapsulation
Solves problems in the design level.	1.	Solves the problem in the implementation level.
Used for hiding the unwanted data and giving relevant data	2.	Hiding the code and data into a single unit to protect the data from outside world.
Allows us to focus on what the object does instead of how it does ~~it.~~ the function.	3.	Encapsulation means hiding the internal details or mechanism of how an object does something
Abstraction - Outer layout, used in terms of design For example:- Outer Look of a Mobile phone, like it has a display screen and keypad buttons to dial a number	4.	Encapsulation - Inner layout, used in terms of implementation. For example:- Inner Implementation detail of a mobile phone, how keypad button and display screen are connected with each other using circuits.
Abstraction done using Interface and Abstract class.	5	Encapsulation done using access modifiers (Public, Protected & Private)

19. What are the conditions to create class, constant, Interface, method, Package, & variable.

Class
- It should start with the uppercase letter
- It should be a noun such as Color, Button, System, thread etc.
- Use appropriate words, instead of acronyms.

Example.

```
public class Employee
{
// code snippet
}
```

Constant
- It should be in uppercase letters such as RED, YELLOW.
- If the name contains multiple words, it should be separated by an underscore(_) such as:- MAX_PRIORITY
- It may contain digits but not as first letter

Example

```
class Employee
{
//constant
static final int MIN_AGE = 18;
// code snippet
}
```

Interface

- It should start with the uppercase letter
- It should be an adjective such as Runnable, Remote, ActionListener.
- Use appropriate words, instead of acronyms.

```
interface Printable
{
//code snippet
}
```

Example.

Method

- Should start with lowercase letter
- It should be a verb such as main(), print(), println()
- If the name contains multiple words, starts with lowercase letter followed by an uppercase letter such as actionPerformed().

```
class Employee
{
//method
void run(){
// code snippet
}
}
```

Package

- It should be a lowercase letter such as java, lang etc.
- If the name contains multiple words, it should be separated by dot (.) such as java.util, java.lang.

```
package com.jjp ;// package
class Employee{
  //code snippet.
  }
```

Variable

- Should start with lowercase such as id, name
- It should not start with special character like & (ampersand), $ (dollar), _ (underscore)
- Do camelCasing. firstName, lastName
- Avoid using one-character such as: x, y, z

```
class Employee
{
//variable
 int id;
  int valueId = 5;
 //code snippet
}.
```

What is abstraction in Java?

Abstraction refers to the quality of dealing with ideas rather than events. It basically deals with hiding the details and showing the essential things to the user. Thus you can say the abstraction in Java is the process of hiding the implementation details from the user and revealing only the functionality to them. Abstraction can be achieved in two ways:-

1. Abstract classes (0-100% of abstraction can be achieved)
2. Interface (100% abstraction can be achieved)

Q: What do you mean by an interface in Java?

An interface in Java is a blueprint of a class or you can say it is a collection of abstract method and static constants. Each method is public and abstract but it does not contain any construction. It is a group of related method with empty bodies.

Example:

```
Public interface Animal{
  Public void eat();
  Public void sleep();
  Public void run();
  }
```

20. What are the difference betⁿ abstract classes & interface ?

Abstract Class		Interface.
An abstract class can contain ~~access~~ both incomplete (abstract) and complete member.	1	An interface contains only incomplete members (signature of members).
Does not support multiple inheritance	2	Supports multiple inheritance.
Abstract class contains Constructors.	3	Interface does not contain constructor.
Only complete members of abstact class can be static	4	Members of interface can not be static
In case of an abstract class, a class may extends only one abstract class	5.	A class may implement several interfaces.
An abstract class can have any visibility: Public, private, protected	6.	An interface visibility must be public.
If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly	7.	If we add a new method to an interface then we have to track down all the implementations of the interface and define implementation for the new method.

21. Why multiple inheritance not supported in Java through class.

If a child inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.

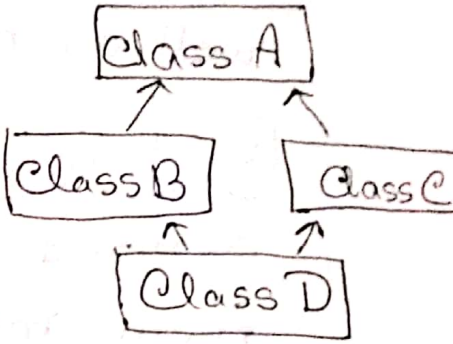


Figure: Diamond problem

The problem with multiple inheritance is that ~~if~~ multiple parent classes may have the same method name, then at run-time it becomes difficult for the complier to decide which method to execute from the child class.

Therefore, Java doesn't support multiple inheritance for classes. The problem is commonly referred to as Diamond Problem.

22. What are different types of inheritance in Java?

Java supports four types of inheritance. Which are:

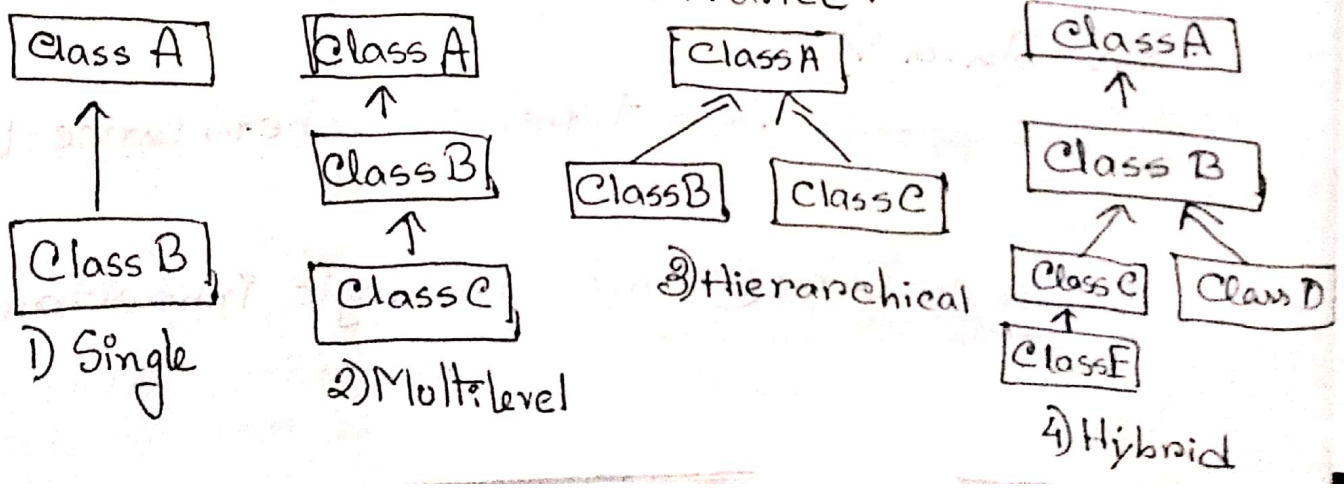
(1) Single Inheritance: In single inheritance, one

class inherits the properties of another i.e. there will be only one parent as well as one child class.

(2.) Multilevel Inheritance: When a class is derived from a class which is also derived from another class. i.e. a class having more than one parent class but at different levels such type of inheritance is called Multilevel Inheritance.

(3.) Hierarchical Inheritance: When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.

(4.) Hybrid Inheritance: It is combination of two or more types of inheritance.



23. What is the static method?

⇒ A static method belong to the class rather than the object.

⇒ There is no need to create the object to call the static method.

⇒ A static method can access and change the value of the static variable.

24. Why is the main method static?

Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation.

25. Can we override the static methods?

No, we can't override static method.

26. Why is inheritance used in Java?

There are various advantages of using inheritance in Java that is given below:-

- Inheritance provides code reusability. The derived class does not to redefine the method of base class unless it needs to provide the

specific implementation of the method.

- Runtime polymorphism cannot be achieved without using inheritance.
- Inheritance provides data hiding. The base class can hide some data from the derived class by making it private.
- Method overriding cannot be achieved without inheritance
- We can simulate the inheritance of classes with real-time objects which makes OOPs more realistic.

~~26~~.27. Why ~~Java~~ does Java not support Pointers?

The pointer is a variable that refers to the memory address. They are not used in Java because they are unsafe (unsecured) and complex to understand.

28. What is super in java ?

The super keyword in Java is a reference variable that is used to refer to the immediate parent class object. Whenever you create the instance of the subclass, an instance of the parent class is created implicitly which is referred by super reference variable. The super() is called in the class constructor implicitly by the compiler if there is no super or this.

Main uses of super keyword :-

- super can be used to refer to the immediate parent class instance variable.
- Super can be used to invoke the immediate parent class method.
- super() can be used to invoke immediate parent class constructor.

30. Can we declare constructor as final ?

The constructor can never be declared as final because it is never inherited. Constructors are not ordinary methods; therefore, there is no sense to declare constructor as final. However if we try to do so, the compiler will throw an error.

31. What is Virtual function in OOP ?

In object-oriented programming, a virtual function or virtual method is a function or method whose behaviour can be overridden within an inheriting class by a function with the same signature to provide the polymorphic behavior.

Therefore according to definition, every non-static method in JAVA is by default virtual method except final and private methods. The methods which cannot be inherited for polymorphic behavior is not virtual method.

32. Difference betn Composition and Aggregation.

Composition		Aggregation
Defines a strong-coupled relationship between two entities, when the one entity is part of another, and both need each other for their existence	1.	Defines a weak-coupled relationship between two entities, where one entity could be part of another, but either can exist without the other, independantly
e.g. Human body and the heart	2.	eg. School & teacher
Composition implies real ownership of its component.	3	Aggregation does not necessarily own any of its aggregates
Composition has a stronger bond of it's components	4	Aggregation has weaker or looser bonds with it's aggregate.
Composition has components that exist at the inner level	5	Aggregation has aggregates that live at the outer level.

33. What is Static Binding & Dynamic Binding?

Static or early binding is resolved at compile time. Method overloading is an example of static binding.

Dynamic or late or virtual binding is resolved at run-time. Method ~~overda~~ overriding is an example of dynamic binding.

34. What is dot (.) Operator ?

The variables methods belonging to a class are formally called member variables and member methods. To reference a member variable or method,

1. First identify the instance.
2. Use dot operator (.) to reference the desired member variable or method.

35. Difference bet^n Compile-time & Run-time Polymorphism?

Compile-time		Runtime
In compile-time, call to a method is resolved at compile-time	1.	In runtime, call to an overridden method is resolved at runtime.
It is also known as static binding, early binding or overloading.	2.	It is also known as dynamic binding, late binding, overriding or dynamic method dispatch.
Overloading is a way to achieve compile-time polymorphism in which we can define multiple methods or constructors with diffent signatures (parameters)	3.	Overriding is a way to achieve runtime polymorphism in which, we redefine some particular method or variable in the derived class.
It provides fast execution because the type of an object is determined at compile-time.	4	It provides slower execution as compare to compile-time because the type of an object is determined at run-time.
Compile-time polymorphism provides less flexibility because all the things are resolved at compile-time.	5	Run-time polymorphism provides more flexibility because all the things are resolved at runtime

36. ~~A method~~ Difference betw Static (class) method and instance method ?

	Static or Class method	Instance method.
1	A method that is declared as static is known as the static method.	A method that is not declared static is known as the instance method.
2	We don't need to create the object to call the static method.	The object is required to call the instance method
3	Non-static (instance) ~~me~~ members cannot be accessed in the static context (static method, block and nested class) directly	Static and non-static variables both can be accessed in instance method.

4	For example:
	Public static int cube(int n)
	{
	return n*n*n;
	}
		For example.
		~~Public void~~
		Public void msg()
		{
		...
		}

37. Why use Java interface?

Mainly 3 reasons to use interface.

- It is used to achieve fully abstraction
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

38. Why java interface allows multiple inheritance?

Java supports multiple inheritance through interfaces only. A class can implement any number of interface but can extend only one class.

Multiple inheritance is ~~not~~ support by interface because there is no ambiguity as implementation is provided by the implementation class. The interface contains all abstract methods and a class can implement multiple interface just because the methods are overridden ~~and imp~~ in ~~them~~ the class.

```
InterfaceA {
  public void hi();
}
```

```
InterfaceB {
  public void hi();
}
```

```
public class myClass
implements InterfaceA,
InterfaceB {
  hi() {
    SOUT ("Hello!");
  }
```

39. Can we make Constructor static?

As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make constructor static However, if you try to do so, the compiler will show the compiler error.

40. Which class is super class of all the classes?

The object class is the superclass of all other classes in Java.

41. Can we execute a program without main() method?

Yes, one of the ways to execute a program without main() method is using static block.

42. Why is inheritance used in Java?

① For code reusability. The derived class does not need to redefine the method of base class unless it needs to provide specific implementation of method.

(2) Runtime polymorphism cannot be acheived without using inheritance

(3) We can simulate the Inheritance of class with real-time objects which makes OOPs more realistic.

(4.) Inheritance can provide data hiding. The base class can hide some data from derived class by making it private.

43. Can we change the scope of the overridden method in subclass?

Yes, we can change scope of overridden method in the subclass. However, we must notice that we cannot decrease the accessibility of method.

- The private can be changed to protected, public or default.
- The protected can be changed to public or default.
- The default can be changed to public
- The public will always remain public

44. Can we declare an interface final?

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition Therefore, there is no sense to make interface final. The compiler will show error if you try.

45. What is difference between final method and abstract method ?

The main difference between final method & abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition.

46. Can you declare an object of abstract class ?

A class that is declared using "abstract" keyword is known as abstract class It can have abstract method (methods without body) as well as concrete methods (regular methods with body) ... An abstract class can not be instantiated which means you are not allowed to create an object of it.

46. How to ~~make~~ make a read-only class in Java?

A class can be made read-only making all of the fields private. The read-only class will have getter methods which return private property of class to main method. We cannot modify this property because there will be no setter method available in class. Example:

```
public class Student{
private String college= "ICE";

public String getCollege (){
    return college;
    }
}
```

47. What are conditions for a class to became abstract?

1) Abstract method has no body.
2) Always end the declaration with a semicolon(;)
3) It must be overridden.

An abstract class must be extended and in a same way abstract ~~m~~method must be overridden

48. What is object class?

Object is one of the special class defined by Java.

All other classes are subclass of Object.

This means tho

Conditions for a class to become abstract?

- Abstract classes may or may not contain abstract methods, i.e. methods without body (public void get();).
- But, if a class has at least one abstract method then the class must be declared abstract
- If a class is declared abstract, it cannot be instantiated.
- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract method in it.

THE END

BEST OF LUCK