OBJECT™
TECHNOLOGIES

OOP Implemetation

What will be covered

OBJECT

- Reference equality
- Association - has-a relationship
- Concept of static
- Use of Array
- Wrapper classes

Reference equality

- When we use the equality operator with primitive types, we re just comparing their values.
- int x, y; //comparing ints

  x = 10;

  y = 15;

  System.out.println(x == y); // prints 'false'

- When == operator is used with reference types, it is used for referential equality comparison i.e. whether two references refer to the same object irrespective of the state of 2 objects.

```
Date d1 = new Date();
Date d3 = d1;
if(d1 == d3)
   System.out.println("d1 and d3 refer to the same object");
else
  System.out.println("d1 and d3 do not refer to the same object");
```

Association - has-a relationship

OBJECT TECHNOLOGIES

- Association is a relationship between two separate classes and the association can be of any type say one to one, one to may etc. It joins two entirely separate entities.
- Composition and Aggregation are the two forms of association.
- Aggregation is a special form of association which is a unidirectional one way relationship between classes (or entities). In Aggregation, both the entries can survive individually which means ending one entity will not effect the other entity.
- In composition, both the entities are dependent on each other.
- When there is a composition between two entities, the composed object cannot exist without the other entity.

Association - has-a relationship - Example

OBJECT

- Class Person maintains information about birthdate of a person by using Date object. Here we are implementing has-a relationship between Person and Date(Person has Date)

```
public class Person
{
    private String name;
    private Date bdate;
    //constructors
    //instance methods
}
```

- bdate is a instance variable of type Date and name is even instance variable of type String. name and bdate both are reference type variables.
- By default all instance variable which are non-primitive(reference) will start the value as null.

Association - has-a relationship - Example

OBJECT TECHNOLOGIES

- Following is the definition of constructors. In the default constructor, it is not necessary to initialize specufically to null (defaultly initialized to null). In the parameterised constructor, using the required data references can be instantiated to appropriate objects. For instantiating bdate dd, mm and yy values are required.

```
public Person()
{
   this.name=null;
   this.bdate=null;
}
public Person(String name,int dd,int mm,int yy)
{
   this.name=name;
   this.bdate=new Date(dd,mm,yy);
}
```

Association - has-a relationship - Example

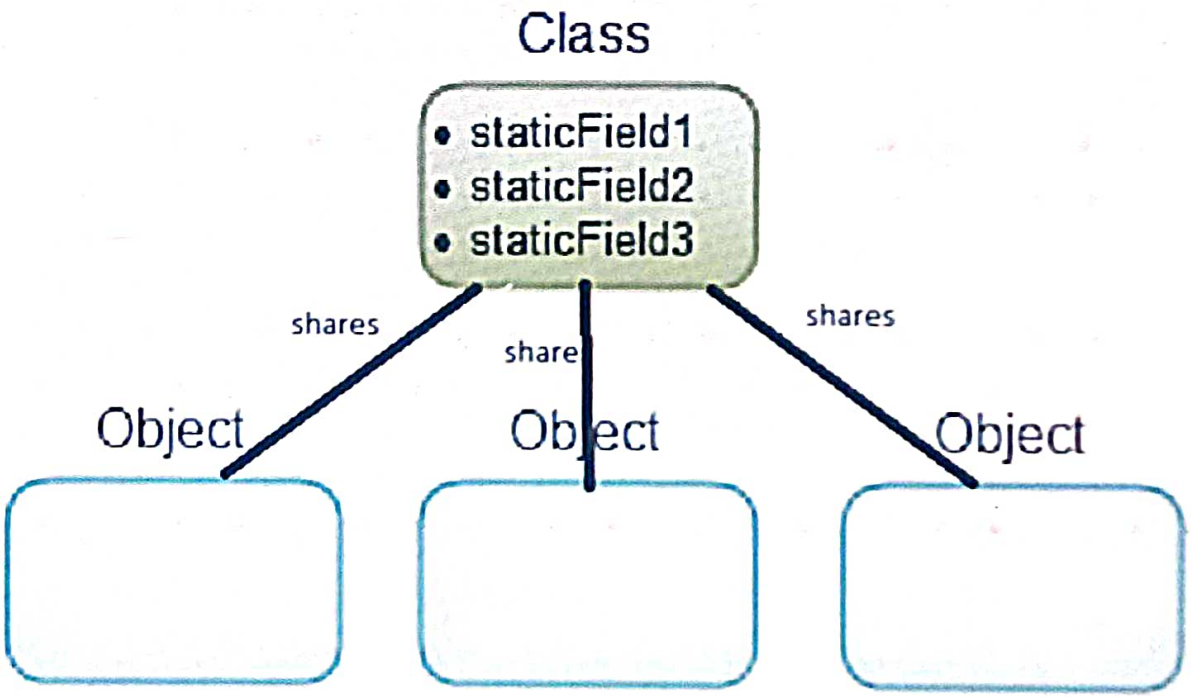
OBJECT TECHNOLOGIES

- In the instance method display, check whether reference variables are not null and then display its information. If this is not checked, it may lead to run time error (NullPointerException).
- Chance of getting this error is mainly when object is created using default constructor.

```
public void display()
{
   System.out.println("Name : "+name);
   if(bdate != null)
   {
      System.out.println("Birth date : ");
      bdate.showDate();
   }
}
```

Concept of static

- In the Java programming language, the keyword static indicates that the particular member belongs to a type itself, rather than to an instance of that type.
- This means that only one instance of that static member is created which is shared across all instances of the class.
- The static keyword can be applied to variables, methods, blocks and nested class.



Concept of static

- Example - If there is class Account which has instance variables declared as acc_no, acc_name and balance. Now there is a need to maintain the information about int_rate. But this int_rate is going to be same for all the instances of class Account.
- Acc_no, acc_name and balance are instance varibales that means these varibales will be maintained separately for each instance of class Account.
- If we maintain int_rate as a instance varibale, unnecessary every instance will have its own copy.
- To avoid this problem, we can declare the int_rate as static. Static means only one copy will be maintained across all instances.

```
public class Account
{
    private int acc_no;
    private String acc_name;
    private double balance;

    private static float int_rate;
    //constructors
    //instance methods
}
```

Static block

OBJECT TECHNOLOGIES

- Static block is used to initialize the static data member.
- It is nothing but pair of curly braces modified with keyword static
- This block gets executed when the class is loaded in the memory. A class gets loaded in memory when it's object is created for the first time.
- A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

```
public class Account
{
    private int acc_no;
    private String acc_name;
    private double balance;
    private static float int_rate;
    static
    {
        int_rate = 3.7f;
    }
```

Static methods

OBJECT TECHNOLOGIES

- When a method is declared with the static keyword, it is known as a static method.
- The most common example of a static method is the main( ) method.
- Methods declared as static can have the following restrictions:
  - They can directly call other static methods only.
  - They can access static data directly.
- Static methods can be called with the help of class name directly. Objects are needed to call the static method. Though static methods can be called with instances, it is discouraged.
- Static methods can not refer this and super (related with inheritance)

Static nested classes

OBJECT

- A class can be made static only if it is a nested class.
- Outer class can never be delared as static.
- Nested static class doesn't need a reference of Outer class. In this case, a static class cannot access non-static members of the Outer class.

Use of Arrays

OBJECT TECHNOLOGIES

- Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type
- In Java all arrays are dynamically allocated using new operator. Every element in the array is recognized by using index which will start with 0. Maximum index available will be length-1
- Since arrays are objects in Java, we can find their length using the object property length.
- Declaring Array Variables
  - dataType[] arrayName;
  - e.g double[] data;
- Creating array instance
  - data = new Double[10];
  - Here data is an array which can store 10 double values

Use of Arrays

OBJECT TECHNOLOGIES

- Array declarion and creation can be clubbed in one statement
- Initializing array
  - e.g int[] age = {12, 4, 5, 2, 5};
  - we have created an array named age and initialized it with the values inside the curly brackets.
  - Note that we have not provided the size of the array. In this case, the Java compiler automatically specifies the size by counting the number of elements in the array (i.e. 5).
  - Alternatively, // declare an array
  - int[] age = new int[5];
  - // initialize array
  - age[0] = 12;
  - age[1] = 4;
  - age[2] = 5; ..

Use of Arrays

OBJECT TECHNOLOGIES

- Accessing array elements :
  - Array elements can be accessed using specific index value

    e.g. System.out.println("First Element: " + age[0]);
  - Using for loop

```
for(int i = 0; i < age.length; i++) {
    System.out.println(age[i]);
}
```

    we are using the length property of the array to get the size of the array.
  - Using for-each loop

```
for(int a : age) {
    System.out.println(a);
}
```

class String

- String is a sequence of characters, for e.g. “Hello” is a string of 5 characters. In java, string is an immutable object which means it is constant and can cannot be changed once it has been created.
- Strings can be created by assigning a String literal to a String instance:

  ```
  String str1 = "Welcome";
  ```

- String can be created using new keyword

  ```
  String str1 = new String("Welcome");
  ```

- String manipulation is one of the most common activities in computer programming. String class has a variety of methods for string manipulation.
- These functionalities include getting the character at perticular index, concatenating, replacing the character, trimming etc

Wrapper classes

- **The wrapper class in Java provides the mechanism to convert primitive into object and object into primitive.**
- **In other words, wrapper classes provide a way to use primitive data types (int, char, short, byte, etc) as objects**
- Need of wrapper classes
  - Wrapper Class will convert primitive data types into objects. The objects are necessary if we wish to modify the arguments passed into the method (because primitive types are passed by value).
  - The classes in java.util package handles only objects and hence wrapper classes help in this case also.
  - Data structures in the Collection framework such as ArrayList and Vector store only the objects (reference types) and not the primitive types.
  - The object is needed to support synchronization in multithreading.
- Note: Primitive types are more efficient than corresponding objects. Hence, when efficiency is the requirement, it is always recommended primitive types.

Wrapper classes

OBJECT

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

All numeric wrapper classes inherit from java.lang.Number

Wrapper classes

- Autoboxing
- The automatic conversion of primitive data types into its wrapper class objects is known as autoboxing.

```
int a = 15; // Primitive data type
Integer I = a; // Autoboxing will occur internally.
```

- Autounboxing
- The automatic conversion of wrapper class objects into its primitive data types is known as unboxing.

```
Integer a = new Integer(15); // Wrapper class object
int I = a;// Unboxing will occur internally.
```

know-it

+ -
Home
Courseware
Practice MCQ
+ Exam
+ Feedback
Profile
Logout

Autoboxing and unboxing lets developers write cleaner code, making it easier to read.

Association Relationship

When an object of one class is created as a data member inside another class, it is called Has-A relationship.

In other words, a relationship in which an object of one class has a reference to an object of another class or another instance of the same class is called Has-A relationship in Java.

Person class has an instance variable of type Address as shown in the below code. An instance of Address class is created outside of Person class.

```
public class Address {
  // Code goes here.
}
public class Person {
  // Person has-a Address.
  Address addr = new Address();
  // Other codes go here.
}
```

Types of Has-A Relationship in Java

There are two types of Has-A relationship that are as follows:

a. Aggregation

b. Composition

Aggregation : Aggregation is one of the core concepts of object-oriented programming. It focuses on establishing a Has-A relationship between two classes.

In other words, two aggregated objects have their own life cycle but one of the objects has an owner of Has-A relationship and child object cannot belong to another parent object.

For example, a library has students. If the library is destroyed, students will exist without library.

Composition : Composition is another one of the core concepts of object-oriented programming. It focuses on establishing a strong Has-A relationship between the two classes.

In other words, two composited objects cannot have their own life cycle. That is, a composite object cannot exist on its own. If one composite object is destroyed, all its parts are also be deleted.

For example, a house can have multiple rooms. A room cannot exist independently and any room cannot belong to two different houses. If the house is destroyed, all its rooms will be automatically destroyed.

Autoboxing and unboxing lets developers write cleaner code, making it easier to read.

Association Relationship

When an object of one class is created as a data member inside another class, it is called Has-A relationship.
In other words, a relationship in which an object of one class has a reference to an object of another class or another instance of the same class is called Has-A relationship in Java.
Person class has an instance variable of type Address as shown in the below code. An instance of Address class is created outside of Person class.

```
public class Address {
  // Code goes here.
}
public class Person {
  // Person has-a Address.
  Address addr = new Address();
  // Other codes go here.
}
```

Types of Has-A Relationship in Java

There are two types of Has-A relationship that are as follows:
a. Aggregation
b. Composition

Aggregation : Aggregation is one of the core concepts of object-oriented programming. It focuses on establishing a Has-A relationship between two classes.
In other words, two aggregated objects have their own life cycle but one of the objects has an owner of Has-A relationship and child object cannot belong to another parent object.
For example, a library has students. If the library is destroyed, students will exist without library.

Composition : Composition is another one of the core concepts of object-oriented programming. It focuses on establishing a strong Has-A relationship between the two classes.
In other words, two composited objects cannot have their own life cycle. That is, a composite object cannot exist on its own. If one composite object is destroyed, all its parts are also be deleted.
For example, a house can have multiple rooms. A room cannot exist independently and any room cannot belong to two different houses. If the house is destroyed, all its rooms will be automatically destroyed.

Association Relationship

When an object of one class is created as a data member inside another class, it is called Has-A relationship.

In other words, a relationship in which an object of one class has a reference to an object of another class or another instance of the same class is called Has-A relationship in Java.

Person class has an instance variable of type Address as shown in the below code. An instance of Address class is created outside of Person class.

```
public class Address {
  // Code goes here.
}
public class Person {
   // Person has-a Address.
   Address addr = new Address();
   // Other codes go here.
}
```

Types of Has-A Relationship in Java

There are two types of Has-A relationship that are as follows:

a. Aggregation

b. Composition

Aggregation : Aggregation is one of the core concepts of object-oriented programming. It focuses on establishing a Has-A relationship between two classes.

In other words, two aggregated objects have their own life cycle but one of the objects has an owner of Has-A relationship and child object cannot belong to another parent object.

For example, a library has students. If the library is destroyed, students will exist without library.

Composition : Composition is another one of the core concepts of object-oriented programming. It focuses on establishing a strong Has-A relationship between the two classes.

In other words, two composited objects cannot have their own life cycle. That is, a composite object cannot exist on its own. If one composite object is destroyed, all its parts are also be deleted.

For example, a house can have multiple rooms. A room cannot exist independently and any room cannot belong to two different houses. If the house is destroyed, all its rooms will be automatically destroyed.

Composition : Composition is another one of the core concepts of object-oriented programming. It focuses on establishing a strong Has-A relationship between the two classes.

In other words, two composited objects cannot have their own life cycle. That is, a composite object cannot exist on its own. If one composite object is destroyed, all its parts are also be deleted.

For example, a house can have multiple rooms. A room cannot exist independently and any room cannot belong to two different houses. If the house is destroyed, all its rooms will be automatically destroyed.

Assignments

1. Create a class Account which maintains information about accno and balance. Add appropriate constructors and display method which displays information about account.
2. Add the static variable as int_rate(float), initialize in the static block. Add the static method updateIntRate() for updating existing int_rate. Display the balance with interest calculated as per int_rate. Modify int_rate and show the balance with int_rate.
3. Create a class named Book to maintain information about bookid, price(float). Add the method to display information about the book. create the book object . Value of id should be assigned on the basis of no of book objects and call its display method. Add default consructor and parameterised constructor which accepts only the price.
4. Modify the Date class to maintain information about the count of Date objects created. Display the count after each Date object created.
5. Create array of 5 integers and assign some values. Find out the max, min and average of this array.
6. Create 3*3 array for storing integers. Create single dimensional arrays to maintain rowwise sum and columnwise sum of the 2d array.
7. Create array of 3 boxes. Display volume calculated for each box. Use class Box which is already created.
8. Create a class person storing name and birthdate of the person (for storing birthdate use user defined date class). Display information about person using display method. Prevent NullPointerException.
9. Create a class Address having data members as area, city and pincode (all are strings). Add appropriate constructors and display method. Create a class Customer having data members as emailid and address. Add the method for display.
10. Create class Line having data members as Start Point and End Point. Add appropriate constructors and display method which should display about start point and end point.
11. Using assignment no 7, accept information about length, width and height from user and display only Box with maximum volume.
12. Create class Student (roll number , name, number of subjects, marks of each subject).No. of subjects varies for each

12. Create class Student (roll number , name, number of subjects, marks of each subject).No. of subjects varies for each student. Write a parameterized constructor which initializes roll number, name, number of subjects and create the array of marks dynamically. Display details of all students.
13. Extend assignment no 12 to calculate percentage of each student and display name and pecentage of Student.
14. Create Class Author with string name, string email, long mobileno.Create class Book with static int bkno, int bookid, Author author, float price, int qty. Write getter ,Setter method and default and fully parametrised constructor in Author and book class. Override toString() method in Author and Book class.
    Write static block to initialise bkno with 1000 and assign its increased value to bookid in Books constructor. Write BookDemo class with main Method which will create array of 5 books.
15. Define class Engine (String type, String fuel ,int cc) and class Car (Engine engine, String make,String clolor).Write getter ,Setter methods, toString() and default and fully parametrised constructor in both the classes.Write CarDemo class with main method which will create 3 different car objects.

Assignments-1

1. Add the static variable as int_rate(float), initialize in the static block. Add the static method updateIntRate() for updating existing int_rate. Display the balance with interest calculated as per int_rate. Modify int_rate and show the balance with int_rate.
2. Create a class named Book to maintain information about bookid, price(float). Add the method to display information about the book. create the book object . Value of id should be assigned on the basis of no of book objects and call its display method. Add default consructor and parameterised constructor which accepts only the price.
3. Modify the Date class to maintain information about the count of Date objects created. Display the count after each Date object created.
4. Create array of 5 integers and assign some values. Find out the max, min and average of this array.
5. Create 3*3 array for storing integers. Create single dimensional arrays to maintain rowwise sum and columnwise sum of the 2d array.
6. Create array of 3 boxes. Display volume calculated for each box. Use class Box which is already created.
7. Create a class person storing name and birthdate of the person (for storing birthdate use user defined date class). Display information about person using display method. Prevent NullPointerException.
8. Create a class Address having data members as area, city and pincode (all are strings). Add appropriate constructors and display method. Create a class Customer having data members as emailid and address. Add the method for display.
9. Create class Line having data members as Start Point and End Point. Add appropriate constructors and display method

9. Create class Line having data members as Start Point and End Point. Add appropriate constructors and display method which should display about start point and end point.

Assignments-2

1. Create a class person storing name and birthdate of the person (for storing birthdate use user defined date class). Accept information about person and display information about person using display method.
2. Create class Line having data members as Start Point and End Point. Add appropriate constructors and display method. Add one more method to calculate the length of the line.
3. Create a class Address having data members as area, city and pincode(all are strings). Add appropriate constructors and display method. Create a class Customer having data members as emailid and address. Add the method for display.
4. Modify above class Person to maintain information about no of person objects created. Add display method to display no of person objects created.
5. Create a class Account having data members as accno, accname and balance. Maintain static data member as int_rate. Provide method to update interest rate and calculate balance with interest.
6. Create a class named Book to maintain information about bookid, title(String), price(float). Add the function to display information about the book. create the book object . Value of id should be assigned by application on the basis of no of book objects and call its display method.
7. Create array of 5 integers and assign some values. Find out the max, min and average of this array
8. Create 3*3 array for storing integers. Create single dimensional arrays to maintain row-wise sum and column wise sum.
9. Create array of 3 boxes. Display volume calculated for each box.
10. Create class Student (roll number , name, number of subjects, marks of each subject).No. of subjects varies for each student. Write a parameterized constructor which initializes roll number, name, number of subjects and create the array of marks dynamically. Display details of all students with percentage and class obtained.

Important Questions for: OOP in Java-III

Interview Questions

Explain OOPs concepts. Went deep into polymorphism(JVM level). Trick question(both interviewers said Overriding is compile time polymorphism).

1) What is OOP?
2) Why OOP is used?
3) What is polymorphism?

9. Create array of 3 boxes. Display volume calculated for each box.
10. Create class Student (roll number , name, number of subjects, marks of each subject).No. of subjects varies for each student. Write a parameterized constructor which initializes roll number, name, number of subjects and create the array of marks dynamically. Display details of all students with percentage and class obtained.

Important Questions for: OOP in Java-III

Interview Questions

Explain OOPs concepts. Went deep into polymorphism(JVM level). Trick question(both interviewers said Overriding is compile time polymorphism).

1) What is OOP?
2) Why OOP is used?
3) What is polymorphism?
4) What is run time and compile time polymorphism?
5) Where and how we use run time and compile time polymorphism?
6) How method overriding is used?
7) Do you know generalization and specialization?
8) What is abstract class?
9) How interface is different from abstract class?
10) When to use abstract class and when to use interface?
11) Program:

Write a program to print sums of even numbers and odd numbers from an array. Use any language.

eg.

arr = [1, 22, 32, 41, 3, 5, 12]

then,

evensum = 22 + 32 + 12

oddsum = 1 + 41 + 3 + 5

Coding Questions

Explain OOPS Concepts with proper technical related example