Biyani Girls College

I Internal Examination-Sept. 2019

Class: - BCA III

Subject: - Core Java Programming (BCA 304)

MM: 40 SET-A Time: 1 ½ Hrs.

[I]Very short answer questions (Max 40 words) (5 * 2 = 10)

1. What is ByteCode?

Ans. Java bytecode is the result of the compilation of a Java program, an intermediate representation of that program which is machine independent.

The Java bytecode gets processed by the Java virtual machine (JVM) instead of the processor. It is the job of the JVM to make the necessary resource calls to the processor in order to run the bytecode.

2. Why java is more secure language as compare to C/C++?

Ans. 1. As java program runs inside its own virtual machine sandbox

2. Data hiding in Java(OOPs) makes it one of the secure language. Maybe some points are also there but cannot recall it right now.
3. No use of pointers preventing unauthorized access to memory block.
4. No access to the memory management
5. Access Control Functionality
6. Exception Handling
7. Package java.security provides the classes and interfaces for the security framework

3. What is Class Variable?

Ans. Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block. There would only be one copy of each class variable per class, regardless of how many objects are created from it.

4. What is difference between compile time and run time binding?

Ans.

BASIS FOR COMPARISON	STATIC BINDING	DYNAMIC BINDING
Event Occurrence	Events occur at compile time are "Static Binding".	Events occur at run time are "Dynamic Binding".
Information	All information needed to call a function is known at compile	All information need to call a function come to

	time.	know at run time.
Time	Fast execution.	Slow execution.
Alternate name	Early Binding.	Late Binding.
Example	Overloaded function call, overloaded operators.	Virtual function in C++, overridden methods in java.

5. What is super class?

Ans. Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

[II]Short answer questions (Max 80 words) (2 * 5 = 10)

1. Define different type of variables with syntax and example?

Ans. A variable is a name given to a memory location. It is the basic unit of storage in a program.

Types of variables:-

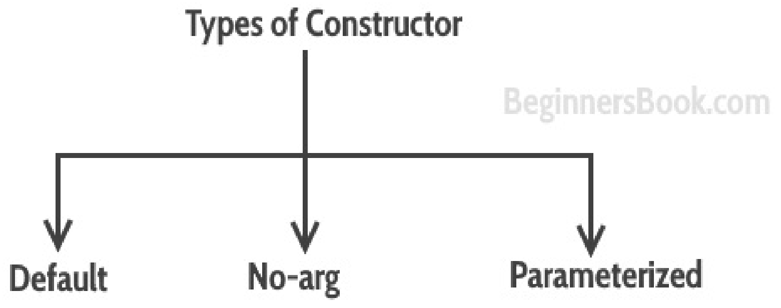
1. **Local Variables**: A variable defined within a block or method or constructor is called local variable.
   i. These variable are created when the block in entered or the function is called and destroyed after exiting from the block or when the call returns from the function.
   ii. The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variable only within that block.
   iii. Initilisation of Local Variable is Mandatory.
2. **Instance Variables**: Instance variables are non-static variables and are declared in a class outside any method, constructor or block.
   1. As instance variables are declared in a class, these variables are created when an object of the class is created and destroyed when the object is destroyed.
   2. Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier then the default access specifier will be used.
   3. Initilisation of Instance Variable is not Mandatory. Its default value is 0
   4. Instance Variable can be accessed only by creating objects.
3. **Static Variables**: Static variables are also known as Class variables.
   - These variables are declared similarly as instance variables, the difference is that static variables are declared using the static keyword within a class outside any method constructor or block.
   - Unlike instance variables, we can only have one copy of a static variable per class irrespective of how many objects we create.

- Static variables are created at the start of program execution and destroyed automatically when execution ends.
- Initilisation of Static Variable is not Mandatory. Its default value is 0

4. Define different type of constructor with example?

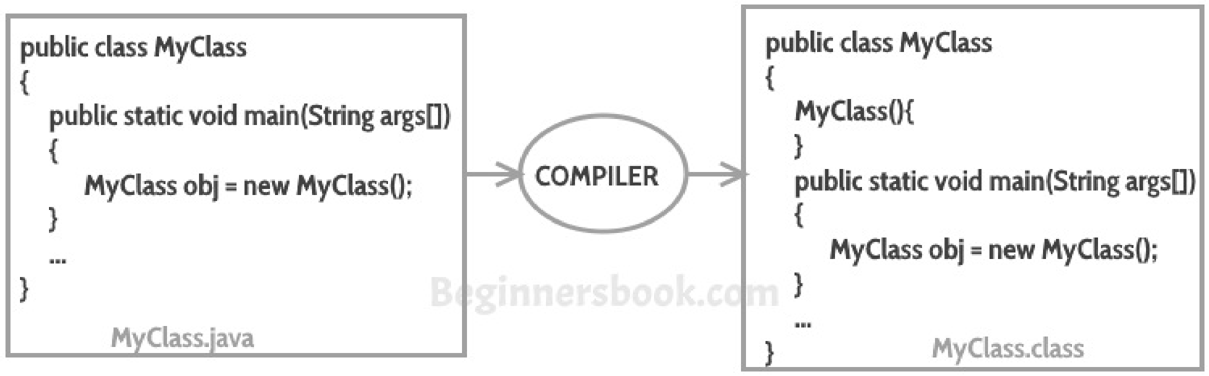
Ans.

There are three types of constructors: Default, No-arg constructor and Parameterized.



1. Default Constructor

If you do not implement any constructor in your class, Java compiler inserts a default constructor into your code on your behalf. This constructor is known as default constructor. You would not find it in your source code(the java file) as it would be inserted into the code during compilation and exists in .class file. This process is shown in the diagram below:



```
public class MyClass
{
    public static void main(String args[])
    {
        MyClass obj = new MyClass();
    }
    ...
}
```

```
public class MyClass
{
    MyClass(){
    }
    public static void main(String args[])
    {
        MyClass obj = new MyClass();
    }
    ...
}
```

If you implement any constructor then you no longer receive a default constructor from Java compiler.

2. no-arg constructor:

Constructor with no arguments is known as **no-arg constructor**. The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.

Although you may see some people claim that that default and no-arg constructor is same but in fact they are not, even if you write **public Demo() { }** in your class `Demo` it cannot be called default constructor since you have written the code of it.

Example: no-arg constructor

```
class Demo
{
    public Demo()
    {
       System.out.println("This is a no argument constructor");
    }
    public static void main(String args[]) {
         new Demo();
    }
}
```

Output:

```
This is a no argument constructor
```

3. Parameterized constructor

Constructor with arguments(or you can say parameters) is known as Parameterized constructor.

Example: parameterized constructor

In this example we have a parameterized constructor with two parameters `id` and `name`. While creating the objects `obj1` and `obj2` I have passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.

```
public class Employee {
   int empId;
   String empName;

   //parameterized constructor with two parameters
   Employee(int id, String name){
       this.empId = id;
       this.empName = name;
   }
   void info(){
        System.out.println("Id: "+empId+" Name: "+empName);
   }

   public static void main(String args[]){
          Employee obj1 = new Employee(10245,"Chaitanya");
          Employee obj2 = new Employee(92232,"Negan");
          obj1.info();
          obj2.info();
   }
}
```

Output:

```
Id: 10245 Name: Chaitanya
Id: 92232 Name: Negan
```

[III]Long answer questions (Max 150 words). **(2 * 10 = 20)**

1. What is inheritance and define types of inheritance in java with example?

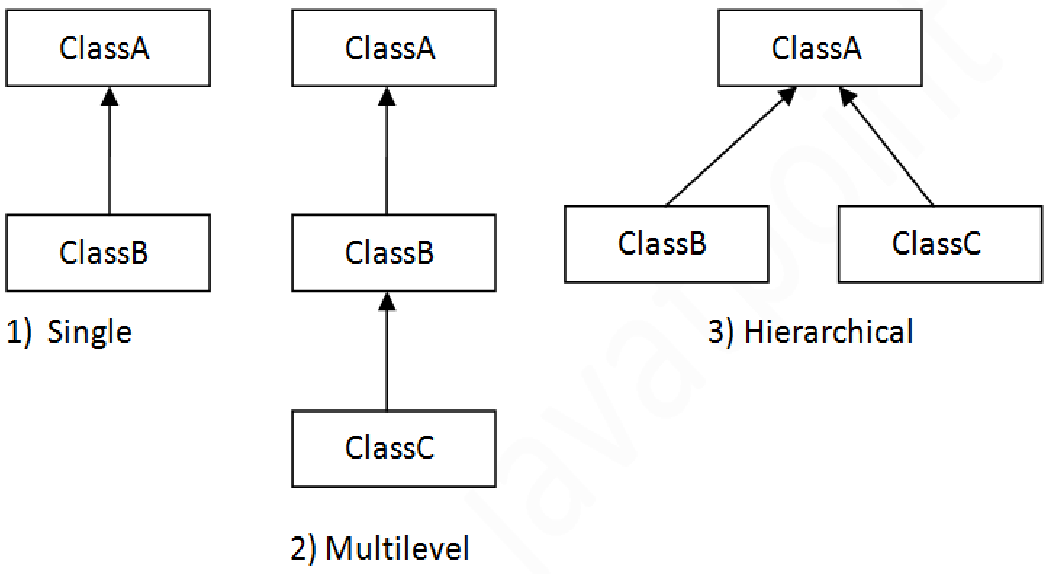
Ans.

Inheritance: Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only.



- **Single Inheritance**: - When a class extends another one class only then we call it a single inheritance.
- **Multilevel Inheritance**:- It refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class.
- **Hierarchical inheritance:**- In such kind of inheritance one class is inherited by many sub classes.

2. Write a program to check a given year is leap or not?

Ans.

```
import java.util.Scanner;
public class Check_Leap_Year
{
    public static void main(String args[])
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter any year:");
        int year = s.nextInt();
        boolean flag = false;
        if(year % 400 == 0)
        {
            flag = true;
        }
        else if (year % 100 == 0)
        {
            flag = false;
        }
        else if(year % 4 == 0)
        {
            flag = true;
        }
        else
        {
            flag = false;
        }
        if(flag)
        {
            System.out.println("Year "+year+" is a Leap Year");
        }
        else
        {
            System.out.println("Year "+year+" is not a Leap Year");
        }
    }
}
```

Biyani Girls College

I Internal Examination(Solution) Sept. 2019

Class: - BCA III

Subject: - Core Java Programming (BCA 304)

MM: 40 Set: B Time: 1 ½ Hrs.

[I] Very short answer questions (Max 40 words). (5 * 2 = 10)

1. Name three types of comments in java?

Ans. Single Line Comment(used to comment only one line)

Multi Line Comment(used to comment multiple lines of code)

Documentation Comment(used to create documentation API)

3. Difference between continue and break statement?

Ans. The break keyword is used to breaks(stopping) a loop execution, which may be a for loop, while loop, do while or for each loop.

The continue keyword is used to skip the particular recursion only in a loop execution, which may be a for loop, while loop, do while or for each loop

4. What is difference between constructor and normal methods?

Ans. Constructor is used to initialize an object whereas method is used to exhibits functionality of an object.

Constructors are invoked implicitly whereas methods are invoked explicitly.

Constructor does not return any value where the method may/may not return a value.

In case constructor is not present, a default constructor is provided by java compiler. In the case of a method, no default method is provided.

Constructor should be of the same name as that of class. Method name should not be of the same name as that of class.

5. What do you understand by package?

Ans. a package is an organized and functionality based set of related interfaces and classes. Packages organize classes that belong to the same category or provide similar functionality.

Package names should be unique, and a package class may access package access members for other package classes.

6. What is finally bock?

Ans. Java finally block is a block that is used to execute important code such as closing connection, stream etc.

Java finally block is always executed whether exception is handled or not.

Java finally block follows try or catch block.

[II] Short answer questions (Max 80 words). (2 * 5 = 10)

1. Give three differences between final and static variable?

Ans.

final –

1)When we apply "**final**" keyword to a **variable**,the value of that variable remains constant. (or) Once we declare a **variable** as **final**.the value of that variable cannot be changed.

2)It is useful when a **variable** value does not change during the life time of a program

3)Syntax

```
final int pie = 3.14;
```

static -

1)when we apply "**static**" keyword to a **variable** ,it means it belongs to class.

2)When we apply "**static**" keyword to a **method**,it means the **method** can be accessed without creating any instance of the class

3)Syntax

```
static int pie = 3.14;
```

2. What is difference between method overloading and method overriding?

Ans
. Here are some important facts about Overriding and Overloading:
1). The real object type in the run-time, not the reference variable's type, determines which overridden method is used at *runtime*. In contrast, reference type determines which overloaded method will be used at *compile time*.
2). Polymorphism applies to overriding, not to overloading.
3). Overriding is a run-time concept while overloading is a compile-time concept.
Example:

1. Overriding

```
class Dog{
            public void bark(){
               System.out.println("woof ");
            }
         }
         class Hound extends Dog{
            public void sniff(){
               System.out.println("sniff ");
            }

            public void bark(){
               System.out.println("bowl");
            }
         }

         public class OverridingTest{
            public static void main(String [] args){
               Dog dog = new Hound();
               dog.bark();
            }
         }
```

2. Overloading

```
class Dog{
   public void bark(){
      System.out.println("woof ");
   }

   //overloading method
   public void bark(int num){
        for(int i=0; i<num; i++)
                  System.out.println("woof ");
   }
}
```

[III] Long answer questions (Max 150 words) (2 * 10 = 20)

1. Define OOPs Concepts in Java?

Ans.

1. Object

Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.
An Object can be 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 message accepted and 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 behaviors like wagging the tail, barking, eating, etc.

2. Class

Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

3. Inheritance

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

4. Polymorphism

If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

5. Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

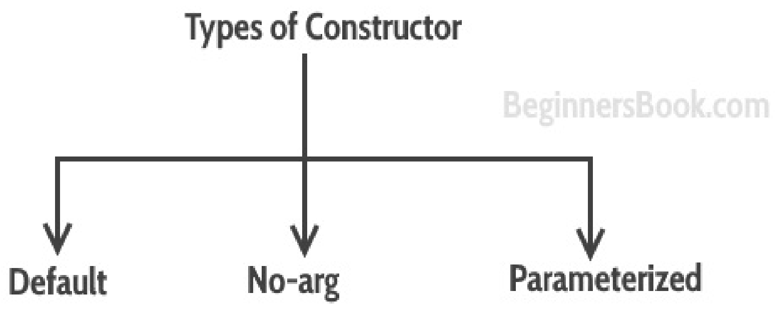
6. Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

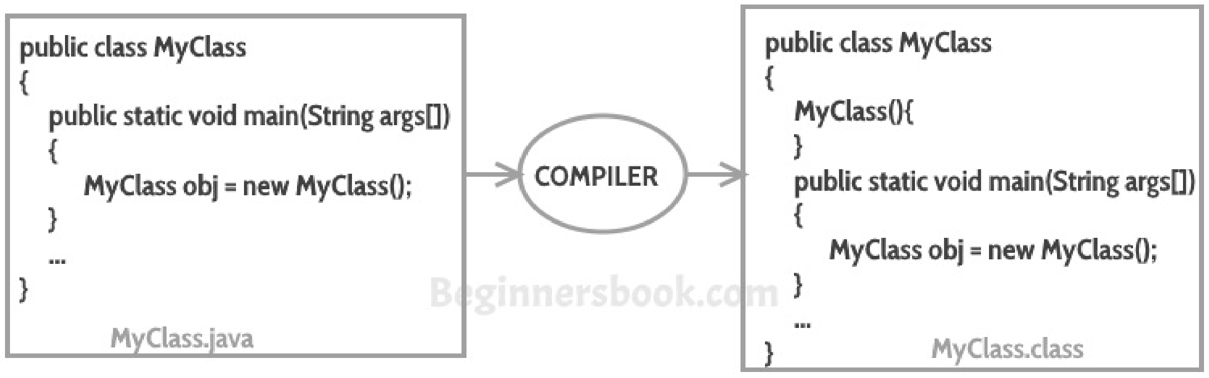
2. Define different types of constructor with example?

Ans. There are three types of constructors: Default, No-arg constructor and Parameterized.



4. Default Constructor

If you do not implement any constructor in your class, Java compiler inserts a default constructor into your code on your behalf. This constructor is known as default constructor. You would not find it in your source code(the java file) as it would be inserted into the code during compilation and exists in .class file. This process is shown in the diagram below:



```
public class MyClass
{
    public static void main(String args[])
    {
        MyClass obj = new MyClass();
    }
    ...
}
```

```
public class MyClass
{
    MyClass(){
    }
    public static void main(String args[])
    {
        MyClass obj = new MyClass();
    }
    ...
}
```

If you implement any constructor then you no longer receive a default constructor from Java compiler.

5. no-arg constructor:

Constructor with no arguments is known as **no-arg constructor**. The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.

Although you may see some people claim that that default and no-arg constructor is same but in fact they are not, even if you write **public Demo() { }** in your class Demo it cannot be called default constructor since you have written the code of it.

Example: no-arg constructor

```
class Demo
{
    public Demo()
    {
        System.out.println("This is a no argument constructor");
    }
    public static void main(String args[]) {
     new Demo();
    }
}
```

Output:

```
This is a no argument constructor
```

6. Parameterized constructor

Constructor with arguments(or you can say parameters) is known as Parameterized constructor.

Example: parameterized constructor

In this example we have a parameterized constructor with two parameters `id` and `name`. While creating the objects `obj1` and `obj2` I have passed two arguments so that this constructor gets invoked after creation of obj1 and obj2.

```
public class Employee {
   int empId;
   String empName;

   //parameterized constructor with two parameters
   Employee(int id, String name){
       this.empId = id;
       this.empName = name;
   }
   void info(){
        System.out.println("Id: "+empId+" Name: "+empName);
   }

   public static void main(String args[]){
     Employee obj1 = new Employee(10245,"Chaitanya");
     Employee obj2 = new Employee(92232,"Negan");
     obj1.info();
     obj2.info();
   }
}
```

Output:

```
    Id: 10245 Name: Chaitanya
    Id: 92232 Name: Negan
```