1. Fill in the blanks.
    a. Calling a default constructor will result in error, if parameterised constructor is already defined
    b. this() denotes non parameterized/default constructor
    c. The syntax used to find the length of an array 'A' in JAVA is A.length

2. Write the output/possible error of the following code segment.

a.
```
public class Test1
{
  int count=10;
  Test1 (int count){
  System.out.println("Count=" + count);
  }
  public static void main(String[] args)
  {
  Test1 t = new Test1();
      }
 }
```

Ans. Error in Test1 t = new Test1();
^
required: int
found: no arguments
reason: actual and formacl argument lists differ in length
1 error

b.
```
class Test2  {
        final int i;
        static double j=10.5;
         public static void main(String[] args){
        System.out.println("j=" + j);
  }
}
```

Ans. error: variable i not initialized
final int i;

c.
```
class Test3{
   int a, b;
   Test3(int a, int b) {
      a = a;
      b = b; }
   void display(){
            System.out.println("a = " + a + "  b = " + b);}
   public static void main(String[] args){
      Test3 object = new Test3(10, 20);
      object.display(); }
```

```
}
```
Ans. a = 0 b = 0

```
d.public class Demo{
      public static void main(String[] arr){
            System.out.println("Hi");
             maina("IIITS");}
      public static void maina(String a){
      System.out.println("Hello "+ a); }
}
```

Ans. Hi
Hello IIITS

```
e. class Main {
    public static void main(String args[]) {
        System.out.println( fun() );
         System.out.println( Main.fun() );
         Main obj = new Main();
        System.out.println(obj.fun());
        }
   static int fun() {
      return 20; }
}
```
Ans. 20
20
20

3. What is the role of a just-in-time (JIT) compiler in JAVA?

Ans. Java compiler translates Java source code into bytecodes that represent the tasks to execute. Bytecodes are executed by the Java Virtual Machine (JVM)—a part of the JDK and the foundation of the Java platform. JVMs typically execute bytecodes using a combination of interpretation and so-called just-in-time (JIT) compilation. A just-in-time (JIT) compiler—known as the Java HotSpot compiler—translates the bytecodes into the underlying computer's machine language. **In order to improve performance, JIT compilers interact with the Java Virtual Machine (JVM) at run time and compile suitable bytecode sequences into native machine code. While using a JIT compiler, the hardware is able to execute the native code, as compared to having the JVM interpret the same sequence of bytecode repeatedly and incurring an overhead for the translation process.**

4. Write a JAVA program to print the following pattern using dynamic allocation in a 2D array.

```
1  2  3  4
1  2  3
1  2
1
```

Ans.
```
class PatternGeneration {
    public static void main(String args[]) {
    int twoD[][] = new int[4][];
    twoD[0] = new int[4];
    twoD[1] = new int[3];
    twoD[2] = new int[2];
    twoD[3] = new int[1];
    int i, j, k = 0;
    for(i=0; i<4; i++)
        for(j=i; j<4; j++) {
        twoD[i][j-i] = (j-i)+1;
         }
    for(i=0; i<4; i++) {
        for(j=i; j<4; j++)
        System.out.print(twoD[i][j-i] + " ");
        System.out.println();}
    }}
```

5. Write the differences between the following.
   a. Constructor and Method

Constructor	Method
A block of code that initialize at the time of creating a new object of the class is called constructor.	A set of statements that performs specific task with and without returning value to the caller is known as method.
It is mainly used for initializing the object.	It is mainly used to reuse the code without writing the code again.
It has no return type. It can or cannot return any value to the caller.	So, it has a return type.
The constructor name will always be the same as the class name.	We can use any name for the method name, such as addRow, addNum and subNumbers etc.
Example is required	Example is required

   b. Default constructor and parameterized constructor

Default Constructor	**Parameterized Constructors**
A default constructor is a 0 argument	The parameterized constructors are the

constructor which contains a no-argument call to the super class constructor.	constructors having a specific number of arguments to be passed.
To assign default values to the newly created objects is the main responsibility of default constructor.	The purpose of a parameterized constructor is to assign user-wanted specific values to the instance variables of different objects.
Compiler writes a default constructor in the code only if the program does not write any constructor in the class.	A parameterized constructor is written explicitly by a programmer.
The access modifier of default constructor is always the same as a class modifier but this rule is applicable only for “public” and “default” modifiers.	The access modifier of default constructor is always the same as a class modifier but this rule is applicable only for “public” and “default” modifiers.
Example is required	Example is required

c. public and default access specifier

Public	**Default**
All the class members declared under the public specifier will be available to everyone.	When no access modifier is specified for a class, method, or data member – It is said to be having the default access modifier by default.
The data members and member functions declared as public can be accessed by other classes and functions too. The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.	default access modifiers are accessible only within the same package.
Example is required	Example is required

d. Instance variable and class variable

Instance Variable	Class Variable
It is basically a class variable without a static modifier and is usually shared by all class instances.	It is basically a static variable that can be declared anywhere at class level with static.

Across different objects, these variables can have different values.	Across different objects, these variables can have only one value.
They are tied to a particular object instance of the class, therefore, the contents of an instance variable are totally independent of one object instance to others.	These variables are not tied to any particular object of the class, therefore, can share across all objects of the class.
Example is required	Example is required

6. Write a java program with the following class: 4 Marks

Class BankAccount:

Fields: UserName (String), Password (String), Account_Number (Integer), Account_Name (String), Account_Balance (double)

Static Field: Total_Number_of_Accounts (Integer): To keep track of the total number of accounts

Constructor: with arguments to initialize BankAccount with a username and password and an automatically generated account number

Methods:

- void updateAccount (String UserName, String OldPassword, String New Password) - to change the password (you should receive the username, old password, and new password and reset the password after authentication)
- Void BalanceUpdate(String UserName, String Password, Double Amount)
- void checkBalance (String Username, String Password) - Print the balance (after proper authentication of username and password)
- void transferMoney (String USername, String Password, BankAccount B, double Amount)- send money to another account receiving account object and amount, after proper authentication using username and password; Record the transaction

  // if you are changing the attributes or functions or number or type of arguments of the given functions, please justify the same

  //Please specify and justify the access specifiers for various attributes and functions

Solution:

```
import java.util.*;//
Class BankAccount
{

// 1 marks for properly defining the fields


Private String UserName;
Private String Password;
Private Static int Total_Number_of_Accounts = 0;
Private int Account_No;
Private double Account_Balance




BankAccount(String UserName, String Password) //1 for constructor
{
this.UserName=UserName;
this.Password=Password;
Total_Number_of_Accounts++;
}



updateAccount(String UserName, String Password, String newPassword)  //0.5 for proper
implementation
{
if(this.UserName.equals(UserName)) && (this.password.equals(Password)) {

this.Password = newPassword;
}

}
```

BalanceUpdate(String UserName, String Password, Double Amount) //0.5 for proper implementation`

```
{
if(this.UserName.equals(UserName)) && (this.password.equals(Password)) {

this.Account_balance=this.Account_balance+ Amount;

}

}
```

TransferMoney(String UserName, String Password, Double Amount, BankAccount B) //1 for proper implementation

```
{

if(this.UserName.equals(UserName)) && (this.password.equals(Password))
{
B.balanceupdate(UserName, Password, Amount)
}

}
}
```