JAVA

ASSIGNMENT - 1

D. Sathvik
2211CS010141
Group - 4

ASSIGNMENT - 1

2211CS010141

1)

a) What is JVM, JDK and JRE

A) JVM:-

A Java Virtual Machine is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

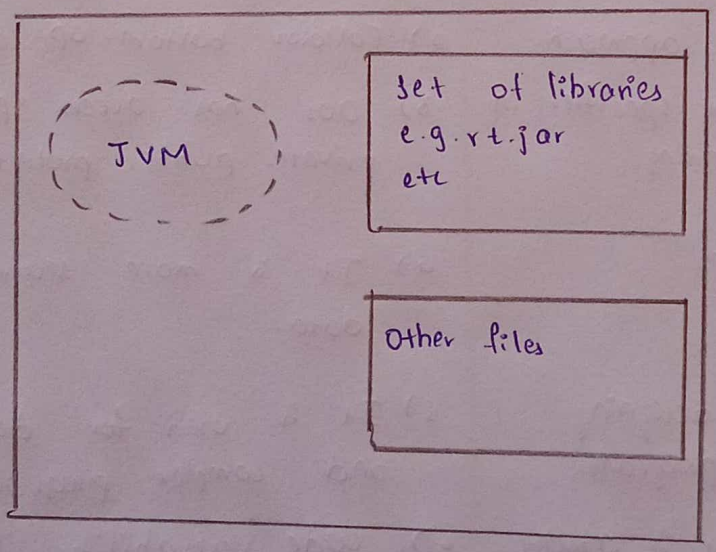
JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent.

JVM performs following main tasks:-

- Loads code
- Verifies code
- Executes code
- Provides runtime environment.

JRE:-

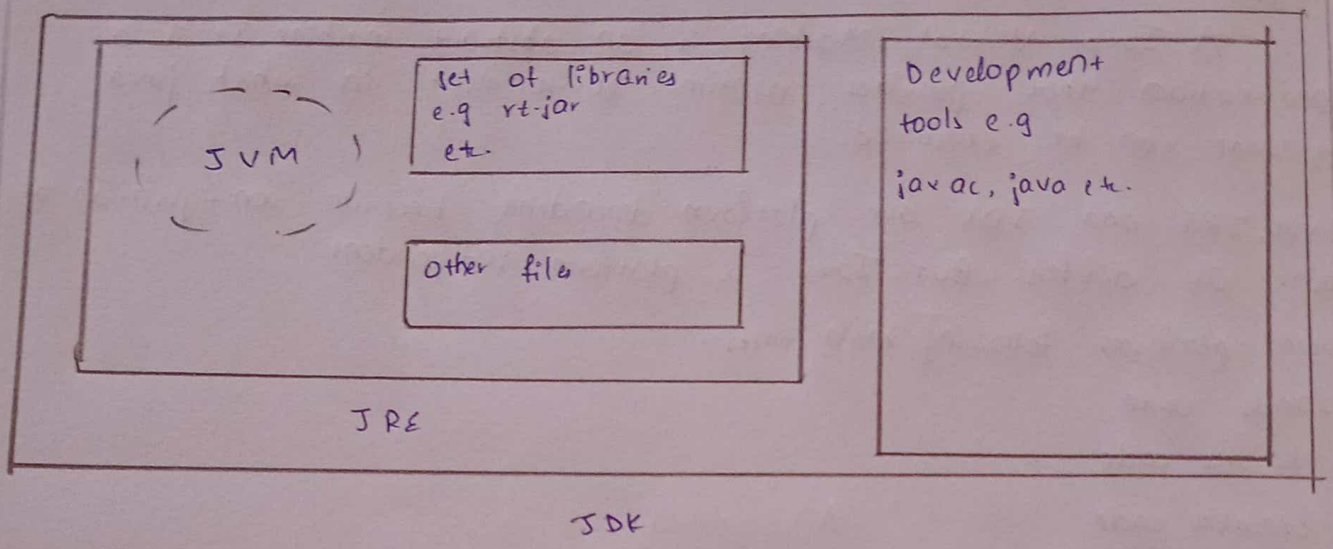
JRE is an acronym for Java Runtime Environment. It is used to provide runtime environment. It is the implementation of JVM. It physically exists. It contains set of libraries + other files that JVM uses at runtime.



A)

JDK:-

JDK is an acronym for Java Development kit. It physically exists. It contains JRE + development tools.



b) Differentiate between Procedure Oriented language and OOPL.

A)

Procedural Oriented Programming	Object-Oriented Programming
1) In procedural programming the program is divided into small parts called functions.	1) In object-oriented programming, the program is divided into small parts called objects.
2) Follows Top-down approach.	2) Follows bottom-up approach.
3) There is no access specifier in procedural programming.	3) OOL has access specifier like private, public, protected, etc.
4) It is less secure.	4) It is more secure and hides data.
5) It is used for designing medium-sized program.	5) It is used for designing large and complex programs.
6) code reusability is absent in procedural programming	6) code reusability is present in OOPL.
Ex:- C, etc.	Ex:- C++, Java, Python, etc.

2) a) What feature of Java makes it platform independent and portable?

A) Java's platform independence and portability are primarily attributed to several key features:

1) Bytecode compilation :-

Java source code is compiled into bytecode rather than native machine code. This bytecode can run on any platform with a JVM which makes it platform independent.

2) Java Virtual Machine :-

The JVM acts as an intermediary b/w the bytecode and the underlying hardware. It's responsible for executing Java programs on various platforms.

3) Write Once, Run Anywhere (WORA) :-

It's WORA principle allows developers to write code on one platform and run it on any other platform with a compatible JVM.

4) Standard Class Library :-

Java provides a standard library with classes and methods that abstract platform-specific details, enhancing the portability of Java applications.

5) Strongly Typed language :-

Java enforces strong typing, which aids in preventing platform-specific data type issues.

6) Exception Handling :-

Java's robust exception handling mechanisms contribute to the stability and reliability of programs, regardless of the platform.

b) Is Java a Robust language? Justify your answer.

A) Java is considered a robust language, and the reasons are:

1) Strong Typing:

Java enforces strong typing, which means it helps catch type-related errors at compile time, reducing runtime errors.

2) Memory Management:

Java incorporates automatic memory management through garbage collection. This helps prevent common programming errors like memory leakage.

3) NO pointers:

Java avoids direct manipulation of memory pointers, which eliminates a common source of bugs.

4) Security Features:

Java includes various security mechanisms which restrict untrusted code from performing harmful operations making it a secure choice.

5) Standard library:

It has a comprehensive standard API that provides a wide range of built-in classes and methods.

6) Multi-threading:

Java's built-in support for multi-threading allows developers to create robust, concurrent applications with ease.

3) a) Differentiate between a class and object.

A)

Object	class
1) Object is an instance of class	1) class is a blueprint from which objects are created.
2) Object is a real world entity such as pen, laptop, etc.	2) Class is a group of similar objects.
3) Object is a physical entity.	3) Class is a logical entity.
4) Object allocates memory when it is created.	4) Class doesn't allocate memory when it is created.
5) There are many ways to create object in java such as new keyword, newInstance() method, clone() method.	5) There is only one way to define class in java using class keyword.

b) Demonstrate constructor overloading concept

A) Java supports constructor overloading in addition to overloading methods. In Java, overload constructor is called based on the parameters specified when a new is executed.

For ex:

The thread class has 8 types of constructor. If we do not want to specify anything about a thread then we can simply use the default constructor.

Each constructor can perform diff initialization based on the parameters provided.

Ex:-

```
Public  clas  Book {
        private string title;
        private string author;
        private int year;

        public  Book (string title, String author){
                this.title = tile;
                this.author = author;
                this.year = 0;
        }
         public  void displayInfo(){
                System.out.println ("Title " + title);
                System.out.println("Author: " + author);
                System.out.println ("Year: " + year);
         }
          public  static  void main (String[] args){
                Book  book1 = new  Book ("To kill a Mockingbird", "Harper
                                                        Lee ");
                Book   book2 = new Book("1984", " Orwell", 1949);

                System.out.println ("Book 1: ");
                book1. displayInfo();

                System.out.println ("Book 2: ");
                book2.displayInfo();
          }
}
```

In this example, Book has 2 constructors title and author as 1 parameters, another title, author and year as one.

4) Explain the basic concepts of object oriented programming.

A) Object oriented programming revolves around the concept of "objects". It is built on several fundamental concepts:

1) Objects :- They are the instances of class. They encapsulate both data and functionality.

2) Classes :- Classes are blueprint or templates for creating objects. They define the structure and behavior that objects of that class will have.

3) Encapsulation :- It is the concept of bundling data and methods that operate on that data into a single unit i.e the object.

4) Inheritance :- Inheritance allows one class to inherit attributes and methods from another class. It promotes code reusability.

5) Polymorphism :- It means that objects of diff classes can be treated as objects of a common superclass. It allows you to write code that works with objects in a more general way.

6) Abstraction :- It is the process of simplifying complex reality by modeling classes based on the essential attributes and behavior.

7) Platform Independence :- Java has the ability of a program or application to run on diff computer platforms without modification.

8) Robustness :- It refers to the ability of a software system to handle errors, invalid inputs, and unexpected situations.

a) Multi threading :- It allows a program to execute multiple threads concurrently.

5) a) What is Type Casting? List and explain types of type casting methods with suitable example.

A)

Type casting, also known as type conversion, is the process of changing an entity's data type from one type to another in a programming language. This is often necessary when you need to perform operations on variables or values of different data types. There are two main types of type casting in Java:

1) Implicit casting (Widening)
2) Explicit casting (Narrowing)

(1) Implicit Casting (Widening):-

Implicit casting occurs when you convert a data type with a smaller range into a data type with a larger range.

Ex:-

```
int x = 10;
double y = x;
```

(2) Explicit Casting (Narrowing):-

Explicit casting is necessary when you convert a data type with a larger range into a data type with a smaller range.

Ex:-
```
double x = 10.5;
int y = (int) x;
```

Widening:-

byte → short → char → int → long → float → double

Narrowing:-

double → float → long → int → char → short → byte.

b) List out primitive data types available in Java and explain?

A) Primitive data types are basic building blocks for representing simple values. The following are

1. byte :-
   Size : 8 bits
   Range : -128 to 127
   Default : 0.

2. Short :
   Size : 16 bit
   Range : - 32,768 to 32,767.
   Default : 0

3. int :-
   Size : 32 bits
   Range : - 2,147,483,648 to 2,147,483,647
   Default : 0

4. long :-
   Size : 64 bit
   Range : - 9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
   Default : 0L

5. float :-
   Size : 32 bits
   Default : 0.0f

6. double :-
   Size : 64 bits
   Default : 0.0d

7. boolean :-
   Size : Not exactly defined
   Values : True or false
   Default : False.

8. char :-
   Size : 16 bits
   Min : \u0000 (or '0')
   Max : \uffff (65,535)

6) Define Constructor. Illustrate various types of Constructor with examples.

A) A constructor in Java is a special method that is automatically called when an object of a class is created. Its primary purpose is to initialize the attributes of the object. Constructors have the same name as the class and do not have a return type.

Types of constructors:-

1) Default Constructor:-

* A default constructor is provided by Java if no constructors are defined in a class.
* It initializes the object with default values (e.g 0 for numeric data type, 'null' for objects).

Ex:-

```
public class MyClass{
}
Myclass obj = new Myclass();
```

2) Parameterized Constructor:-

* A parameterized constructor accepts parameters and initializes the object's state based on the provided values.

Ex:-

```
public class Person{
        private String name;
        private int age;

        public Person (String name, int age){
                this.name = name;
                this.age = age;
        }
}
Person person = new Person ("Alice", 25);
```

3) Copy Constructor :-

* A copy constructor creates a new object as a copy of an existing object. It is used to create a deep copy of an object.

Ex :-

```
public class Book{
    private String title;
    private String author;

    public Book(Book otherBook){
        this.title = otherBook.title;
        this.author = otherBook.author;
    }
}
```

---

7) Explain about control statements in Java with example.

A) It has 3 types of control flow statements.

1) Decision Making statement.
   - if
   - switch

2) Loop statements
   - do while
   - while
   - for
   - for-each

3) Jump statements
   - break
   - continue.

1) Decision Making statement :-

a) if statement :-

It is used to execute a block of code only if a spaified condition is true.

Ex :-

```
int num = 10;
if (num > 5){
    System.out.println("the number is greater than 5.");
}
```

b) if - else statement:

It allows you to execute one block of code if a condition is true and ~~false~~ another block if it is false.

Ex:-
```
int num = 3;
if (num > 5){
    System.out.println ("The number is greater than 5.");
} else {
    System.out.println ("The number is not greater than 5.");
}
```

c) if - else if statement:

It is an extension of if-else statement and is used when you have multiple conditions to check.

Ex:-
```
int num = 7;
if (num < 5){
    System.out.println ("No. is less than 5.");
} else if (num < 10){
    System.out.println ("No. is less than 10, greater than 5");
} else {
    System.out.println ("No. is greater than or equal to 10".)
}
```

2) Loop statements:-

Looping statements allow you to execute a block of code repeatedly

a) for loop:-

The for loop is used to iterate over a range of values.

Ex:-
```
for (int i=0; i<5; i++){
    System.out.println ("Iteration: " +i);
}
```

b) while loop :-

The while loop is used to repeatedly execute a block of code as long as a condition is true.

Ex :-

```
int count = 0;
while (count < 3){
    System.out.println("count : " + count);
    count ++;
}
```

c) do - while loop :-

It is similar to while loop but guarantees that block of code is executed at least once, even if the condition is initially false.

Ex :-

```
int x = 0;
do {
    System.out.println("x : " +x);
    x ++;
} while (x < 3);
```

d) for - each loop :-

It is used to iterate over elements without the need for explicit indexing.

Ex :-

```
int [] numbers = { 1,2,3,4,5 };
for (int num : numbers){
        System.out.println(num);
}
```

3) Jump Statements:

Jump statements allow you to change the normal flow of program execution.

a) break statement:

The break statement is used to exit from a loop prematurely or to terminate a 'switch' statement.

Ex:-

```
for (int i=0; i<5; i++){
    if (i==3){
        break;
    }
    System.out.println("Iteration " + i);
}
```

b) Continue statement:-

It is used to skip the current iteration of a loop and proceed to the next iteration.

Ex:-

```
for (int i=0; i<5; i++){
    if (i==2){
        continue;
    }
    System.out.println("Iteration: " + i);
}
```

8) Describe different levels of access specifiers in Java.

A) In Java, access specifiers, also known as access modifiers, are keywords that determine the visibility and accessibility of classes, fields, methods and constructors within a program. There are four diff levels of access specifier in Java:-

1) Public:-

* The public access specifier provides high level of visibility
* Members declared as public can be accessed from any class and package.
* It allows unrestricted access to the member.

Ex:-

```
public class Myclass {
    public int publicField;
    public void publicMethod() {
        // code
    }
}
```

2) Protected:-

* The protected access specifier restricts access to within the same package and by subclasses.
* It is commonly used for providing controlled access to internals of a class of inheritance and polymorphism.

Ex:-

```
class MyBaseclass {
    protected int protectedField;
    protected void protectedMethod() {
    }
}
class MySubclass extends MyBaseclass {
    void accessBaseMembers() {
        protectedField = 10;
        protectedMethod();
    }
}
```

3) Default :

* The default access specifier (no modifier) allows access only within the same package.
* Members with default access are not accessible from classes in diff packages.

Ex :

```
class MyClass {
    int defaultField;
    void defaultMethod(){
        // code
    }
}
```

4) Private :

* The private access specifier provides the most restrictive level of visibility.
* Members declared as 'private' are accessible only within the same class.
* It is used for encapsulation, allowing you to hide the internal details of a class.

Ex :

```
public class MyClass {
    private int privateField;
    private void privateMethod(){
    // code
    }
}
```

* —— * —— *