1. Introduction

Q.5.2.1.1 Name some programming approaches practised over time.
Answer: Examples of some programming approaches are object oriented programming, modular programming, structured programming, and top-down programming.

Q.5.2.1.2 List a few programming languages that support object oriented programming features.
Answer: A list of programming languages is given as follows: C++, Objective C, Small talk, Java, Ada and Object Pascal.

Q.5.2.1.3 How does C++ differ from Java?
Answer: Some prominent differences between these objected-oriented programming languages are given below:
C++ is a procedural language with an extension to object-oriented programming. Java is a pure object-oriented programming language.
Java does not support pointers, templates, unions, operator overloading, structures. But, C++ supports these features being an extension to C language.
C++ supports multiple inheritance. Java does not support it directly.

Q.5.2.1.4 Identify three essential features of object-oriented programming (OOP).
Answer: Three most important features of OOP are encapsulation, inheritance and polymorphism.
Encapsulation refers to putting data and functions (methods) in a single unit called class. These functions only are allowed to modify the data contained in the object. *Inheritance* is a process of acquiring properties of an object of a super-class by another object of a sub-class. *Polymorphism* is an ability to perform, say an operation, in different ways in different contexts.

Q.5.2.1.5 Fill in the blank.
The basic runtime entities in an object-oriented system are ——— .
Answer: objects

Q.5.2.1.6 How does OOP come into existence?

Answer: Data were independent of the procedures; programmers have to keep track of functions and the way they modify data. Let us first understand the drawbacks of structured programming. The dependency on data and task are too hard to maintain. It is difficult to extend an existing large program. To address these problems of structured programming, OOP came into existence.

Q.5.2.1.7 How does an object communicate with another object?
Answer: In OOP, objects communicate with one another by sending and receiving information using messages. For an object, a message is a request for execution of a method. When the request is made to an object, it invokes the specified method in the receiving object, then the receiving object responds by generating the desired result.

Q.5.2.1.8 Object oriented programming was motivated to address one of the following major concerns of software development. Choose the appropriate one.
(a) modularity, (b) symbolic names in assembly language programs, (c) re-usability of code, (d) large memory requirements of procedural programming.
Answer: (c)

Q.5.2.1.9 Differentiate between object-oriented programming and procedure oriented programming.
Answer: Some important differences between procedure oriented programming (POP) and object- oriented programming (OOP) are given below:
(i) In POP, a program is divided into functions. But a program in OOP is divided into objects.
(ii) In POP, importance is not given to data but to functions as well as sequence of actions to be done. In OOP, importance is given to the data rather than procedures or functions because it works as a real world.
(iii) POP does not have the concept of access specifier. OOP has different access specifiers such as public, private and protected.
(iv) In POP, data can move freely from function to function in the system. In OOP, objects can move and communicate with each other through member functions.
(v) To add new data and functions in POP is not so easy. OOP provides an easy way to add new data and functions.
(vi) Overloading is not possible in POP. In OOP, overloading is possible

in the form of function overloading and operator overloading.
(vii) POP does not have any proper way for hiding data. So it is less secure. OOP provides data hiding and it provides more security.
(viii) A few example of POP are C, VB, FORTRAN and Pascal. Some examples of OOP are C++, JAVA, VB.NET and C#.NET.

Q.5.2.1.10 What are the benefits of using object-oriented programming?
Answer: There are many benefits of object oriented programming. Some benefits are stated below.
1. Through inheritance we can eliminate redundant code and extend the use of existing classes.
2. Object-oriented systems can be easily upgraded from a small system to a large system.
3. Software complexity can be easily managed.
4. It is possible to have multiple instances of an object to co-exist without any interference.
5. Message passing technique for communication between objects makes the interface descriptions with external systems much simple.
6. The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of program.
7. It is easy to partition the work in a project based on objects.
8. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.

2. Basics of Java

Q.5.2.2.1 Write the output of the following program segments.
(i) `float f1 = 4.22F;`
`int x1 = (int) f1;`
`System.out.println("x1 = " + x1);`

(ii) `int i = 888888;`
`short s = (short) i;`
`System.out.println("s = " + s);`

Answer: (i) Output: `x1 = 4`. Fraction part is truncated.

(ii) It executes. But it produces incorrect result because of the following statement.
`short s = (short) i ;`
Value of `i`, i.e., `888888` can not be shortened.

Q.5.2.2.2 Identify correct/incorrect assignments.
(a) `float y = 24.31;`
(b) `char c1 = "5";`
(c) `short s1 = 1234;`
Answer: (a) incorrect (`24.31` is considered as double by default)
(b) incorrect ("`5`" is considered as string constant)
(c) correct

Q.5.2.2.3 How does an instance variable differ from class variable?
Answer: An instance variable of a class has a separate value for each instance of the class. But a class variable represents a single data item shared by the class and all the instances of the class.
A class variable is defined by the keyword static along with the data type of the variable as shown in the following example.

```
class xyz {
   int i;          // instance variable
   static int j;   // class variable
   String s;
}
```

Q.5.2.2.4 Identify the ranges of the short type and character (char) type data.
Answer: Two bytes are allocated to these types of data. The range of the short type is -2^{15} to $2^{15}-1$. In case of character type data, there is no concept of negative character. Thus, the range of the character type is 0 to $2^{16}-1$.

Q.5.2.2.5 Compare and contrast primitive data type `int` with the wrapper class Integer.
Answer: Integer is a class defined in `java.lang` package. `int` is a primitive data type defined in Java language. We take an example to differentiate them.
Consider the following code written in Java.

```
int i = 2;
Integer intObj = new Integer(i);
int j = intObj.intValue();
```

First two lines converts an integer value to an `Integer` object. Last two lines do the opposite to this.

Q.5.2.2.6 Which of the following is not a valid data type in Java?
(a) void, (b) int, (c) Integer, (d) static
Answer: (d)

Q.5.2.2.7 What is the difference between the `rint`() and the `round`() function in Java?
Answer: The `java.lang.Math.rint(double a)` returns the double value that is closest in value to the argument and is equal to a mathematical integer. The `java.lang.Math.round(float a)` returns the closest integer to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type integer. For example, 2.50 lies between 2.00 and 3.00. `Math.rint`() returns the closest even double value. `Math.rint(2.50)` returns 2.0. But, Math.round() returns the closest higher whole number. `Math.round(2.50)` returns 3.

Q.5.2.2.8 Discuss different forms of Java applications.
Answer: Various types of applications can be developed using Java.
(a) *Stand-Alone Applications*
(i) Console Applications: They run from a command promt. They do not display any GUI based screens.

(ii) GUI Applications: They run as a stand-alone application. They accept user input through GUI based screen.
(b) *Web Applications*
These applications require a browser to execute.
(i) Applets: These are programs that run on a Java enabled web browser. Appletviewer is provided to ensure that these programs run without a web browser.
(ii) Servlets: Java is suitable for web-based n-tier application development. In a web-based application, client sends a request, and server processes the request and a response is sent back to the client.The server side Java Application Programming Interfaces (APIs)takes care of program process and the response to the request of the client.
(c) *Distributed Applications*
It requires a server to run these applications. A number of servers can be used for backup to prevent data losses.
(i) Database Applications: These programs use JDBC API for database connectivity.
(d) *Client-server Applications*
These applications use web technologies for their execution. They follow client-server model of computing.

Q.5.2.2.9 Find the answers.
(i) `17 % 3` = ———
(ii) `17 % -3` = ———
(iii) `-17 % 3` = ———
(iv) `-17 % -3` = ———
Answer: (i) 2, (ii) 2, (iii) -2, (iv) -2
Sign of remainder is the sign of first operand.

3. Classes, Objects and Methods

Q.5.2.3.1 What does keyword `static` refer to?
Answer: A member in a class with `static` type means the member belongs to the entire class and is not a part of any object of that class. The following program illustrates the concept of static variable.

```
class Student{
   int rollnum;
   String name;
   static String college = "Chowgule College";
   Student(int r, String n){
   rollnum = r;
   name = n;
   }
   void display (){
     System.out.println(rollnum + " " + name + " "
      + college);
   }
 }
public class TestStudent {
  public static void main(String[] args) {
      Student k = new Student (100, "Kaushik");
      Student m = new Student (200,"Mukul");
      k.display();
      m.display();
   }
```

The output of the above program is given below:
`100 Kaushik Chowgule College`
`200 Mukul Chowgule College`
Both the objects `k` and `m` have the same value for the variable `college`. The variable `college` is a class variable and available to all objects of class `Student`. The static member `college` can be accessed by `Student.college`.

Q.5.2.3.2 What is command line argument? Give an example.
Answer: The `main`() method of Java has the following signature.
`public static void main (String args[ ])`

Here, `args` is declared as an array of String objects. During interpretation (i.e., execution time), one can pass arguments that the program needs to be process. Those arguments are called command line arguments.
Example: Suppose the `main`() method belongs to class House. At the time of running the program one can pass 3BHK as argument stated as follows:
java House 3BHK
Program can process this argument using `args` [`0`].

Q.5.2.3.3 What is default constructor? When does it become useful?
Answer: A constructor is a method having the same name as the class name. By calling a constructor one can create an object and utilize the variables in the object. A constructor without any parameter is called a default constructor that initializes object variables with the default values. An example is shown below.
Example 1: Class Circle with a default constructor is given below:

```
class Circle {
   int radius;
   // default constructor
   Circle () {
      radius = 1;
   }
   // parameterized constructor
   Circle (int x) {
      radius = x;
   }
   float Area () {
      return (3.14F * radius * radius);
   }
}
```

Q.5.2.3.4 How does a static member differ from other members in a class?
Answer: A static member in a class is declared using keyword **static** before the type of the member. For example, two static members are declared as follows.
`static int i`; // static data member of the class
`static float circle` (`float r`); // static method of the class
But an instance member can be declared without the keyword `static`.

For example, an instance data member member `j` can be defined as follows:
`int j;` // instance data member of the class
For different objects, `j` may have different values.
A static member is not part of any object of the class. Rather, it is a member of the class. Static variables can be called class variable and static methods are called class methods. A static method can be called in association with the class it belongs to. If the above members belong to `XYZ` class then one can call the method `circle` in the following way:
`float a1` = `XYZ.circle`(4.0);
Let `obj` be an object of class `XYZ`. Then instance variable `i` can be assigned as follows:
`obj.i` = 20;

Q.5.2.3.5 How does the constructor of a subclass creates instance variables of superclass?
Answer: The subclass constructor uses the keyword `super` to invoke the constructor of superclass. `super` can be used within the construction of subclass, and has parameters of type and order as the same as that of variables declared in the superclass.

```
class Animal {
  Animal() {
    System.out.println("An animal is created");
  }
}
class Tiger extends Animal {
  Tiger() {
    super();
    System.out.println("A tiger is created");
  }
}
class SuperTest {
  public static void main (String args[] ) {
    Tiger t = new Tiger();
  }
}
```

The output of the program is given below:

```
An animal is created
A tiger is created
```