Ref No: EX/IT/T/221/2019

BE in INFORMATION TECHNOLOGY

2ND YEAR, 2ND SEMESTER EXAMINATION, 2019

OBJECT ORIENTED SYSTEMS

Full marks: 100 **Time: 3 hours**

CO1:

1.a) Show with a suitable program how the *main()* method can be overloaded.
Can we overload two methods having the signatures as ***public double sum(int a, double b)*** and ***public static double sum (int x, double y)***? If yes, show how it can be done using suitable code snippet; if not, provide supporting reasons for it.

b) What is the purpose of using interfaces in Java?
Create an interface *GIN* with an abstract method *display()*. Now create 2 more interfaces *IN1* and *IN2* extending *GIN*. Finally define a class "*Sample*" implementing these 2 interfaces *IN1* and *IN2*. Show how the *display()* method of the *GIN* interface can be invoked through an object of the "*Sample*" class.

c) How can you create custom exception classes? Show with suitable examples how to use/extend them to solve your own purpose.

d) Create a Java class "*Time*" with three member variables *hour, minute* and *second.* Define suitable default and parameterized constructors to initialize the members. Now define 2 methods *add(A,B,C)* and *sub(A,B,C)* that take the three members *hour(A), minute(B)* and *second(C)* of *Time* object as parameters and perform the addition and subtraction of those 2 objects. Example given below:
4hr 50min 40sec + 3hr 12 min 30 sec= 8hr 3min 10sec
4hr 10min 40sec -2hr 12 min 50 sec= 1hr 57min 50sec

e) Discuss the functionalities of any 3 methods that provide added flexibilities to the *StringBuffer* class compared to the *String* class.

[(2+1+2)+(2+3)+(3+2)+7+3=25]

Or,

2.a) Assume a class having a default constructor and a parameterized constructor. Show with a suitable code snippet how the parameterized constructor can be called from the body of the default constructor. Provide a code snippet to show how a member method of a class returns the reference of its object.

b) Assume a class *Sample.* Consider the following two scenarios of creating objects.
Show diagrammatically how in memory the references point to the objects that are created for the two cases below.

```
Sample s1=new Sample()
Sample s2;
s2=s1;
Sample s3=new Sample();
s1=s3:
```

```
Sample s1=new Sample(),s2;
Sample s3=new Sample();
s2=s3;
s2=new Sample();
s3=s1:
```

c) Create a Java class "*Employee*" having a member variable *basic_sal* and a member method *salary().* Define a *putData()* method of this class to initialize *basic_sal* with some value taken from the user. Now create another two classes "*Manager*" and "*Clerk*" inherited from "*Employee*" class. Assuming the "*Manager*" and "*Clerk*" get 40% and 20% allowance respectively over the *basic_sal*, override the method *salary()* within the inherited classes to find the *total salary* of a manager and a clerk.

d) Create an abstract class "*Base*" with a default constructor and an abstract method *display()*.Inherit this "*Base*" class to create a "*Derived*" class with its own constructor. Override the *display()* method of the "*Base*" class here. Create one more class "*GrandChild*" inheriting the "*Derived*" class. Override the *display()* method here also. Now call the *display()* method and the parameterized constructors of both the "*Base*" and "*Derived*" classes through an object of the "*GrandChild*" class.

e) How can you use the = = operator for *String* objects?

[(3+3)+(3+3)+6+6+1=25]

<u>CO2:</u>

3.a) Explain the various ways of creating threads in java with suitable examples of code snippets. Discuss the functionalities of *sleep()* and *wait()* methods with its various overloaded versions.

b) What is the purpose of declaring a block as *synchronized*?

Explain with a suitable code how deadlock can occur in a multi-threading environment. Also provide the way of its prevention.

c) How the drawbacks of *Scanner* class can be eliminated? How can you take the input of a character variable using *Scanner* class?

d) Write a Java class "*Student*" having member variables *name, roll, stream* and *cgpa* of string, integer, string and double types respectively. Now find out the *name* and *roll* of the student who has got the highest *cgpa* in IT *stream* out of a total of 400 students of CSE, Electrical, Mechanical, Pharmaceutical and IT streams. Use *BufferedReader* class to take the input of the data members of the students from user.

[(5+2)+(2+5+2)+(2+2)+5=25]

Or,

4.a) Discuss the functionalities of *join()* and *notify()* methods of thread. Show how can you invoke the *run()* method of a thread.

b) Write a short note on thread priority.

c) Discuss the functionalities of *DataInputStream, DataOutputStream, FileInputStream, FileOutputStream* and *InputStreamReader* classes.

d) State the functionalities of *getAbsolutePath(), isDirectory(), read()* and *canRead()* methods.

e) Write a Java class having a method that will copy a text file into another empty file.

[(3+2)+5+5+4+6=25]

<u>CO3:</u>

5.a) Show with a suitable program where arrays of different datatypes (int, float, char) are passed to a generic function. Assume that the function sorts the elements of the array using Bubble Sort technique.

b) Show how you can define a static generic function taking any type of parameter as argument. Also show how it can be invoked from the *main()* method with different types of parameters.

c) Define a generic class with two different types of member variables and having a parameterized constructor that initialized the data members with some value. Show how this constructor can be invoked.

d) Use suitable code snippet to show how you can access the forbidden fields and forbidden methods of a class using reflection. Also show (with the same snippet) how a final field of a class can be modified.

[6+(3+3)+(3+3)+(2+3+2)=25]

Or,

6.a) Assume a class having two public member methods and one private member method. Show with proper code snippet how these member methods of the class can be invoked using reflection.
b) Write suitable code snippet to invoke the *length()* method of *String* class using reflection.
c) Discuss the various ways of creating instances of a class using reflection. Also show how the public member variables of a class can be accessed outside the class definition in reflection.
d) Define a generic class with a member method *find(A,P)* that take sorted arrays *A* of different datatypes (such as int, float, char) as input and checks whether the element ***P*** is present (and at what position) within the array or not using binary search technique.
e) Discuss the application components of Remote Method Invocation.

[3+3+(6+3)+7+3=25]

CO4:

7.a) Distinguish between *include* and *extend* relation of a use case diagram with suitable examples.
b) Identify the actors and the use cases of a library management system. Then draw the use case diagram.
c) Show the interaction among a caller, phone and a recipient in a sequence diagram when a phone call is made.

[5+5+5=15]

Or,

8.a) Show how the various components of a class are represented in a class diagram. What is multiplicity?
b) Distinguish between aggregation and composition relation of a class diagram with proper examples. Show how inheritance is indicated in a class diagram.
c) Identify the entities participating in a university management system and show how they are associated in a class diagram.

[(3+2)+(3+2)+5=15]

CO5:

9. a) Discuss the essential elements of design pattern.
b) Explain command pattern with suitable examples.

[3+7=10]

Or,

10.a) Explain Iterator pattern with proper examples.
b) Discuss singleton design pattern.

[5+5=10]

Course Outcomes:

CO1: Differentiate different object oriented programming language and Solve problems by developing Java programs using (i) classes, (ii) inheritance, (iii) nested classes and (iv) Exceptions.

CO2: Solve problems using thread programming and Input-Output.

CO3: Develop programs using advanced programming paradigms: (i) Introspection capabilities, (ii) Generic Programming.

CO4: Model and Sketch software systems by using different artifacts of Unified Modeling Language.

CO5: Explain and illustrate basics of design patterns by developing programs.