

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY

(Accredited by NAAC with A+ Grade)

Department of Computer Science and Engineering (Accredited by NBA)

Continuous Internal Evaluation Test 1 - AY 2022-23



Course Title : Advanced Java & J2EE		Course Code: 18CS644	
Date: 03/06/2023	Time: 3.00 PM- 04.30 PM	Semester/Section: VI A,B & C	
Faculty: Mr.Senthilkumar R / Dr.Madhusudhan S		Max. Marks: 30	

Note: Answer ONE FULL question from each Module.

Q. No.	Questions		Marks	COs	BTL
Module 1					
1	a)	Discover the enumeration usage with values() and valueOf() methods with an example program.	5	CO1	L3
	b)	Relate the following methods of java.lang.Enum with an example program 1. ordinal 2. compareTo() 3. equals()	10	CO1	L3
OR					
2	a)	How to use wrapper class as numeric type wrapper with an example program in Java and explain various types wrapper used in Java.	10	CO1	L3
	b)	Apply autoboxing or unboxing concept using Java program with proper example.	5	CO1	L3
Module 2					
3	a)	Use Annotation concepts with built in annotations with java program as an example @Override, @inherited, @Retention	10	CO2	L3
	b)	Relate different types of retention policies for annotations in Java with syntax of each annotations	5	CO2	L3
OR					
4	a)	Discover the various methods for various collection frameworks by collection interface with syntax of each interfaces.	10	CO2	L3
	b)	Use the following collection interface: i) Queue ii) SortedSet with syntax for each.	5	CO2	L3

Levels of Bloom's Taxonomy

No.	L1	L2	L3	L4	L5	L6
Level	Remember	Understand	Apply	Analyze	Evaluate	Create

Course Outcomes

CO1	Identify the need for advanced Java concepts like Enumerations and Collections
CO2	Construct client-server applications using Java socket API
CO3	Make use of JDBC to access database through Java Programs
CO4	Adapt servlets to build server side programs
CO5	Demonstrate the use of JavaBeans to develop component-based Java software

QUESTION PAPER REVIEW REPORT

Continuous Internal Evaluation (CIE) Test: 1 AY 2022-23

Department: Computer Science and Engineering Accredited by NBA) Semester/Section: VI / A, B & C

Max Marks: 30

Course Title: Advance Java and J2EE

Course Code: 18CS644

Date: 03/03/2023

Faculty: Dr.Madhusudhan S / Senthilkumar R

Qn. No.	Course Outcome (CO)	Bloom's Taxonomy Level	Marks
1a	1	3	5
1b	1	3	10
2a	1	3	10
2b	1	3	5
3a	2	3	10
3b	2	3	5
4a	2	3	10
4b	2	3	5
Total Marks			60

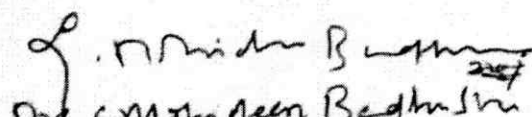
BT Level: L1-Remember, L2-Understand, L3 -Apply, L4 -Analyze, L5- Evaluate, L6- Create

Consolidated Marks for Different BT Levels:

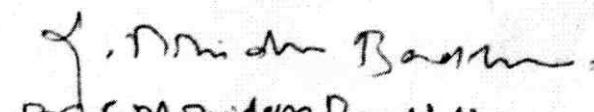
BT Level	Marks for Each Level	% of Marks	Remarks
3	60	60	

Scrutinizer/Reviewer Remark:

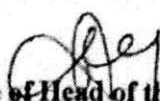
Approved	<input checked="" type="checkbox"/>	Approved with Correction	<input checked="" type="checkbox"/>	Rejected	<input type="checkbox"/>
Reason for Rejection					


Dr. S. M. S. Senthil Kumar
Name & Signature of the Scrutinizer

Date: 24/5/23


Dr. S. M. S. Senthil Kumar
Name & Signature of the IQAC Coordinator

Date: 24/5/23


Signature of Head of the Department
H. O. D.
Dept. Of Computer Science & Engineering
Alva's Institute of Engg. & Technology
Mijar, MOODEJIDRI - 574 228

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY

(Accredited by NAAC with A+ Grade)

Department of Computer Science and Engineering (Accredited by NBA)

Continuous Internal Evaluation Test 1 - AY 2022-23

Course Title : Advanced Java & J2EE	Course Code: 18CS644
Date: 03/06/2023	Time: 3.00 PM- 04.30 PM
Faculty: Mr.Senthilkumar R / Dr.Madhusudhan S	Semester/Section: VI A,B & C
	Max. Marks: 30

Note: Answer ONE FULL question from each Module.

Q. No.	Questions	Marks	COs	BTL
Module 1				
1	<p>a) Discover the enumeration usage with values() and valueOf() methods with an example program.</p> <ul style="list-style-type: none"> The values() method returns an array of enumeration constants. The syntax is, <pre>public static enum_type [] values()</pre> <ul style="list-style-type: none"> The valueOf() method returns the enumeration constants whose value corresponds to the string passed to it as argument. The syntax is. <pre>public static enum_type valuesOf(String str)</pre>	5	CO1	L3
	<p>b) Relate the following methods of java.lang.Enum with an example program</p> <p>1. Ordinal</p> <pre>enum Level { LOW, MEDIUM, HIGH }</pre> <pre>public class Demo{ public static void main(String[] args) { Level myVar = Level.MEDIUM; System.out.println(myVar.ordinal()); } }</pre> <p>2. compareTo()</p> <ul style="list-style-type: none"> The compareTo() is used to compare two ordinal values of the same enumeration. <p>Syntax :</p> <pre>final int compareTo(enum-type e)</pre> <p>For example:</p> <pre>f1 = Fruit.Mangoes; f2 = Fruit.Guavas; f1.compareTo(f2)</pre>	10	CO1	L3

About Enumeration - 2
values & valueOf - 3
with syntax

methods - 2
Program - 6
Program Explain } - 2


```

3. equals()
enum Fruit
{
    Mangoes,
    Guavas,
    banana,
    Oranges,
    Apples;
}

public class EnumDemoInherits
{
    public static void main(String args[])
    {
        Fruit f1,f2,f3;
        System.out.println("All the constants and their ordinal values
are as follows...");

        Fruit allfruits[] = Fruit.values();
        for(Fruit f : allfruits)
            System.out.println(f + ":" + f.ordinal());
        f1 = Fruit.Mangoes;
        f2 = Fruit.Guavas;
        f3 = Fruit.Mangoes;
        if(f1.compareTo(f2)<0)
            System.out.println(f1 + "appear before" + f2);
        if(f2.compareTo(f1)>0)
            System.out.println(f2 + "appear after" + f1);
        if(f1.compareTo(f3) == 0)
            System.out.println(f1 + "equal to" + f3);
        if(f1.equals(f3))
            System.out.println(f1 + "equal to" + f3);
    }
}

```

OR

- 2 a) How to use wrapper class as numeric type wrapper with an example program in Java and explain various types wrapper used in Java.
- The ordinal() method returns the ordinal value of invoking constant. The ordinal value begins with zero.
 - Syntax
- final int ordinal()
- For Example
- ```

Enum Fruit
{
 Mangoes,
 Guaves,

```

About wrapper class - 2  
 Syntax - 2  
 Program - 4  
 Explain - 2

10

CO1

L3

|                 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                         |     |    |
|-----------------|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|-----|----|
|                 |    | banana,<br>Oranges,<br>Apples;<br>}<br>Here, Mangoes has ordinal value , Guavas has ordinal value 1 and so on.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                         |     |    |
|                 | b) | Apply autoboxing or unboxing concept using Java program with proper example. <ul style="list-style-type: none"> <li><b>Autoboxing:</b> Autoboxing is the process by which a primitive type(int, double, float) is automatically encapsulated(boxed) into its equivalent type wrapper(Integer, Double, Float). There is no need to explicitly construct an object. This is an automatic process.</li> <li><b>Definition of Auto-unboxing:</b> Auto-unboxing is process by which the value of boxed object is automatically extracted(unboxed) into respective data type. There is no need to invoke the methods such as <code>intValue()</code> or <code>doubleValue()</code>. This is an automatic process.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Autoboxing - 2<br>unboxing - 2<br>Explain - 1<br>5      | CO1 | L3 |
| <b>Module 2</b> |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                         |     |    |
| 3               | a) | Use Annotation concepts with built in annotations with java program as an example <code>@Override</code> , <code>@Inherited</code> , <code>@Retention</code> <ul style="list-style-type: none"> <li>There are Eight built in annotations. These are defined as follows:<br/> <code>@Retention</code>: It is designed to annotate the other annotation. It is useful for specifying the retention policy.</li> </ul> <p><code>@Override</code>: This annotation is used to inform the compiler that the overriding method is being used from super class. This annotation applies to method only. If we use <code>@Override</code> annotation and the method signature is not found at super class then it will result in compilation error.</p> <p><code>@Override</code>: This annotation is used to inform the compiler that the overriding method is being used from super class. This annotation applies to method only. If we use <code>@Override</code> annotation and the method signature is not found at super class then it will result in compilation error.</p> <p><code>@Inherited</code>: The <code>@Inherited</code> annotation signals that a custom annotation used in a class should be inherited by all of its sub classes. We can use the <code>@Inherited</code> annotation to make our annotation propagate from an annotated class to its subclasses.</p> | 10<br>Each Methods } - 6<br>Each Concepts - 4<br>Syntax | CO2 | L3 |
|                 | b) | Relate different types of retention policies for annotations in Java with syntax of each annotations                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Each Policies - 3<br>Syntax - 2                         | CO2 | L3 |



| Retention Policy        | Availability                                                                                                                                                           |  |  |  |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|
| RetentionPolicy.SOURCE  | The annotation with retention policy of SOURCE is retained only in the source file and discarded during compilation                                                    |  |  |  |
| RetentionPolicy.CLASS   | An annotation with retention policy of CLASS is stored in the .class file during compilation. However it is <b>not available</b> through the JVM <b>during runtime</b> |  |  |  |
| RetentionPolicy.RUNTIME | An annotation with retention policy of CLASS is stored in the .class file and is <b>available</b> through the JVM <b>during runtime</b>                                |  |  |  |

OR

4

a)

Discover the various methods for various collection frameworks by collection interface with syntax of each interfaces.

- Collection classes provide the implementations of different collection interfaces
- The collection is a base interface. It generic interface that has following declaration

interface Collection<E>

Where, E specifies the type of object that the collection will hold.

- Collection extends Iterable interface
- The collection interface contain several useful methods by which we can modify the collections.

| Method                                | Description                                                                         |
|---------------------------------------|-------------------------------------------------------------------------------------|
| Boolean add(Object obj)               | Objects are added using this method.<br>This method takes arguments of type object. |
| Boolean addAll(Collection collection) | Entire contents of one collection can be added to another using this method.        |
| Void clear()                          | In order to clear the collection, this method                                       |

Each method - 6  
Syntax - 2  
Explanation - 2

10

CO2

L3



|                                                   |                                                                                                                                                                    |  |  |  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|
|                                                   | is used.                                                                                                                                                           |  |  |  |
| Boolean<br>contains(Object<br>obj)                | For checking whether the collection contains<br>specific object or not this method returns true.                                                                   |  |  |  |
| Boolean<br>containsAll(Collecti<br>on collection) | If all the elements of the collection are present<br>in the collection then this method returns true.                                                              |  |  |  |
| Boolean isEmpty()                                 | It's a Boolean method which determines<br>whether collection is empty or not.                                                                                      |  |  |  |
| Iterator iterator()                               | Returns iterator to the collection                                                                                                                                 |  |  |  |
| boolean<br>remove(Object obj)                     | An object can be removed by this method                                                                                                                            |  |  |  |
| Boolean<br>removeAll(Collecti<br>on collection)   | It helps in removing a group of objects                                                                                                                            |  |  |  |
| int size()                                        | It returns the total number of elements in the<br>collection                                                                                                       |  |  |  |
| Object[] to Array()                               | Returns an array of elements to the invoking<br>collection. Basically these array elements are<br>the copies of the collection that calls them to<br>Array method. |  |  |  |
| Boolean<br>equals(Object obj)                     | For comparing two collections this method<br>is used.                                                                                                              |  |  |  |

b) Use the following collection interface: i) Queue ii) SortedSet with syntax for each.

**Definition:** The ~~priority~~ <sup>linear</sup> queue is a kind of data structure in which the elements can be inserted in any fashion but while removing the ~~both ends~~ <sup>one end</sup> performed in FIFO order. ~~elements, the high priority elements get deleted first and then the lower priority elements get removed.~~ (2) we define a queue to be a list in which <sup>CO2</sup> all additions to the list are made at one end and all deletions from the list are made at other end.

- Sometimes while inserting the elements in priority queue they get inserted in orderly manner. In such a situation deleting an element becomes a straight forward job.
- This interface is inherited from the set interface and allows the elements to be arranged in ascending order.

L3

Queue (2)  
SortedSet (2)  
Syntax (1)



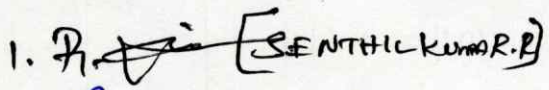
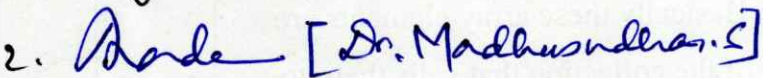
- The methods defined in this interface normally throw the exception such as **NoSuchElementException**, **NullPointerException** and **ClassCastException**.

### Levels of Bloom's Taxonomy

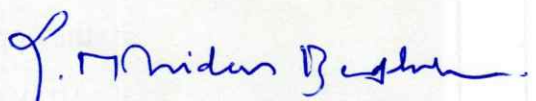
| No.   | L1       | L2         | L3    | L4      | L5       | L6     |
|-------|----------|------------|-------|---------|----------|--------|
| Level | Remember | Understand | Apply | Analyze | Evaluate | Create |

### Course Outcomes

|     |                                                                                |
|-----|--------------------------------------------------------------------------------|
| CO1 | Identify the need for advanced Java concepts like Enumerations and Collections |
| CO2 | Construct client-server applications using Java socket API                     |
| CO3 | Make use of JDBC to access database through Java Programs                      |
| CO4 | Adapt servlets to build server side programs                                   |
| CO5 | Demonstrate the use of JavaBeans to develop component-based Java software      |

1.  [Senthil Kumar B.]
2.  [Dr. Madhusudhan S.]

Faculty Incharge

  
IQAC Co-ordinator

  
HOD-CSE