

**DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING**

**SCHEME**

|  |  |  |  |
| --- | --- | --- | --- |
| *Internal Assessment* | **II** | *Academic Year/Semester* | **2024-25 / ODD S3** |
| *Course Code and Name* | CST 205 OBJECT ORIENTED PROGRAMMING USING JAVA | *Branch* | **CSE** |
| *Date of Exam* | 13/9/2024 | *Duration* | **90 Min** |
| *Starting time* | 9.30 am | *Max. Marks* | **50** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PART-A (*Answer all questions, Each carries 5 marks*) Max Marks: 20** | | | | |
| ***Q.No*** |  | ***Marks*** | ***CO*** | ***Level*** |
| **1** | * cked at compile time. Example: IOException, SQLException. * **Unchecked exceptions**: Exceptions that are not checked at compile time. Example: NullPointerException, ArrayIndexOutOfBoundsException. | **5** | III | L2 |
| **2** | * **Private**: Accessible only within the same class. * **Default**: Accessible within the same package, but not in different packages. * **Protected**: Accessible within the same package and in subclasses across packages. * **Public**: Accessible from any class in any package. | **5** | III | L2 |
| **3** | * equals(): Compares the content of two strings. * compareTo(): Compares lexicographically, returns 0 if equal, a positive number if the first string is greater, and a negative number if it is lesser. | **5** | IV | L2 |
| **4** | add(E e): Adds the specified element to the list.   * remove(Object o): Removes the first occurrence of the specified element from the list. * get(int index): Returns the element at the specified position in the list. | **5** | IV | L2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PART-B (*Each question carries 15 marks)* Max Marks: 30** | | | | | |
| **5** | a. | **try**: Used to enclose code that might throw an exception.   * **catch**: Used to handle the exception thrown by the try block. * **finally**: Block of code that will execute whether an exception is handled or not. * **throw**: Used to explicitly throw an exception. * **throws**: Declares an exception in a method signature that might be thrown.   try {  int result = 10 / 0;  } catch (ArithmeticException e) {  System.out.println("Division by zero error");  } finally {  System.out.println("Finally block executed");  } | **8** | III | L2 |
| b. | An interface in Java is a reference type, similar to a class, that can contain abstract methods, static methods, and constants.   * Example of using an interface:   interface Animal {  void sound();  }  class Dog implements Animal {  public void sound() {  System.out.println("Bark");  }  } | **7** | III | L3 |
| **OR** | | | | | |
| **6** | a | class EvenNumberException extends Exception { }  class OddNumberException extends Exception { }  class NumberChecker {  void checkNumber(int num) throws EvenNumberException, OddNumberException {  if (num % 2 == 0) {  throw new EvenNumberException();  } else {  throw new OddNumberException();  }  }  } | **8** | III | L2 |
| b | A package is a namespace that organizes a set of related classes and interfaces.   * **Creating a package**: Use the package keyword at the beginning of a class. * **Using a package**: Import classes from a package using the import keyword.   package mypackage;  class MyClass { } | **7** | III | L3 |
|  | | | | | |
| **7** | a | Example solution would involve implementing Runnable for both threads, one writing odd numbers to a file, and another writing even numbers to a different file | **8** | IV | L3 |
| b | Mouse events are handled in Java by implementing the MouseListener or MouseAdapter.  Example:  public class MyMouseEvent implements MouseListener {  public void mouseClicked(MouseEvent e) {  System.out.println("Mouse clicked");  }  // Implement other methods (mouseEntered, mouseExited, etc.)  } | **7** | IV | L2 |
| **OR** | | | | | |
| **8** | a | Example solution would use Thread or Runnable to create threads that print odd and even numbers, managing the range and output formatting. | **8** | IV | L3 |
| b | n the Delegation Event Model, the event source generates events, and registered listeners handle those events. Event listeners are registered to the event source using methods like addActionListener(). | **7** | IV | L2 |