**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **13/06/2020** | **Name:** | **Lavanya B** |
| **Course:** | **JAVA** | **USN:** | **4al17ec043** |
| **Topic:** | **Programming core JAVA** | **Semester & Section:** | **6th A** |
| **Github Repository:** | **Lavanya-B** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **Interfaces**  **An interface is a completely abstract class that contains only abstract methods.**  **Some specifications for interfaces:**  **- Defined using the interface keyword.**  **- May contain only static final variables.**  **- Cannot contain a constructor because interfaces cannot be instantiated.**  **- Interfaces can extend other interfaces.**  **- A class can implement any number of interfaces.**  **Eg:**  **interface Animal {**  **public void eat();**  **public void makeSound();**  **}**  **class Cat implements Animal {**  **public void makeSound() {**  **System.out.println("Meow");**  **}**  **public void eat() {**  **System.out.println("omnomnom");**  **}**  **}**  **public class Program {**  **public static void main(String[] args) {**  **Cat c = new Cat();**  **c.eat();**  **}**  **}**  **Casting**  **Assigning a value of one type to a variable of another type is known as Type Casting.**  **To cast a value to a specific type, place the type in parentheses and position it in front of the value.**  **Eg:**  **public class Program {**  **public static void main(String[] args) {**  **double a = 42.571;**  **int b = (int) a;**  **System.out.println(b);**  **}**  **}**  **Anonymous classes**  **Anonymous classes are a way to extend the existing classes on the fly. The modification is applicable only to the current object, and not the class itself. So if we create another object of that class, the start method's implementation will be the one defined in the class.**  **Eg:**  **class Machine {**  **public void start() {**  **System.out.println("Starting...");**  **}**  **}**  **class Program {**  **public static void main(String[ ] args) {**  **Machine m1 = new Machine() {**  **@Override public void start() {**  **System.out.println("Wooooo");**  **}**  **};**  **Machine m2 = new Machine();**  **m2.start();**  **}**  **}**  **Inner class**  **Java supports nesting classes; a class can be a member of another class.**  **Creating an inner class is quite simple. Just write a class within a class. Unlike a class, an inner class can be private. Once you declare an inner class private, it cannot be accessed from an object outside the class.**  **Eg:**  **class Robot {**  **int id;**  **Robot(int i) {**  **id = i;**  **Brain b = new Brain();**  **b.think();**  **}**  **private class Brain {**  **public void think() {**  **System.out.println(id + " is thinking");**  **}**  **}**  **}**  **public class Program {**  **public static void main(String[] args) {**  **Robot r = new Robot(1);**  **}**  **}** |