

ASSIGNMENT

B. LIKHITHA

Assignment 1: Person → Student

Create:

- Person class → name, age
- Student class → rollNo
- Display all details using child object

Concepts: Basic inheritance

The screenshot shows a Java code editor interface. On the left, the code for `Person.java` is displayed. It contains two classes: `Person` and `Student`. The `Person` class has fields `name` and `age`, and a `main` method that creates a `Student` object, sets its name to "Likhitha", age to 20, and roll number to 101, then calls its `display` method. The `Student` class extends `Person` and overrides the `display` method to print the name, age, and roll number. On the right, the "Output" panel shows the execution results: "Name: Likhitha", "Age: 20", "Roll No: 101", and "==== Code Execution Successful ===".

```
Person.java
1- class Person {
2     String name;
3     int age;
4
5     public static void main(String[] args) {
6         Student s = new Student();
7         s.name = "Likhitha";
8         s.age = 20;
9         s.rollNo = 101;
10        s.display();
11    }
12 }
13- class Student extends Person {
14     int rollNo;
15     void display() {
16         System.out.println("Name: " + name);
17         System.out.println("Age: " + age);
18         System.out.println("Roll No: " + rollNo);
19     }
20 }
```

Name: Likhitha
Age: 20
Roll No: 101
==== Code Execution Successful ===

Assignment 2: Vehicle → Car

Create:

- Vehicle → speed
- Car → brand
- Use inherited variable in child

Concepts: Property inheritance

The screenshot shows a Java code editor with the following code in `Vehicle.java`:

```
1- class Vehicle {
2-     int speed;
3-     public static void main(String[] args) {
4-         Car c = new Car();
5-         c.brand = "BMW";
6-         c.speed = 220;
7-
8-         c.showDetails();
9-     }
10- }
11
12- class Car extends Vehicle {
13-     String brand;
14-
15-     void showDetails() {
16-         System.out.println("Brand: " + brand);
17-         System.out.println("Speed: " + speed);
18-     }
19- }
```

The output window shows:

```
Brand: BMW
Speed: 220
==== Code Execution Successful ===
```

Assignment 3: Animal → Dog

Create:

- Animal → eat()
- Dog → bark()
- Call both methods using Dog object

Concepts: Method inheritance

The screenshot shows a Java code editor with the following code in `Animal.java`:

```
1- class Animal {
2-     void eat() {
3-         System.out.println("Animal is eating...");
4-     }
5-
6-     public static void main(String[] args) {
7-         Dog d = new Dog();
8-         d.eat();
9-         d.bark();
10-    }
11- }
12
13- class Dog extends Animal {
14-     void bark() {
15-         System.out.println("Dog is barking...");
16-     }
17- }
```

The output window shows:

```
Animal is eating...
Dog is barking...
==== Code Execution Successful ===
```

Assignment 4: Employee → Manager

Create:

- Employee → salary
- Manager → bonus
- Calculate total salary

Concepts: Code reuse

```

Employee.java
1~ class Employee {
2~     double salary;
3~     public static void main(String[] args) {
4~         Manager m = new Manager();
5~         m.salary = 30000;
6~         m.bonus = 10000;
7~         m.showTotalSalary();
8~     }
9~ }
10~ class Manager extends Employee {
11~     double bonus;
12~     void showTotalSalary() {
13~         double total = salary + bonus;
14~         System.out.println("Salary: " + salary);
15~         System.out.println("Bonus: " + bonus);
16~         System.out.println("Total Salary: " + total);
17~     }
18~ }
19~ 
```

Output

```

Salary: 30000.0
Bonus: 10000.0
Total Salary: 40000.0
== Code Execution Successful ==

```

Assignment 5: Bank → SavingAccount

Create:

- Bank → interestRate
- SavingAccount → calculateInterest()

Concepts: Parent data usage

```

Bank.java
1~ class Bank {
2~     double interestRate = 5.5;
3~     public static void main(String[] args) {
4~         SavingAccount s = new SavingAccount();
5~         double interest = s.calculateInterest(10000);
6~         System.out.println("Interest Rate: " + s.interestRate);
7~         System.out.println("Interest on 10000 = " + interest);
8~     }
9~ }
10~ class SavingAccount extends Bank {
11~     double calculateInterest(double amount) {
12~         return (amount * interestRate) / 100;
13~     }
14~ }
15~ 
```

Output

```

Interest Rate: 5.5
Interest on 10000 = 550.0
== Code Execution Successful ==

```

Assignment 6: Constructor Inheritance

Create:

- Person constructor initializes name
- Student constructor initializes rollNo
- Use super() to call parent constructor

Concepts: super() keyword

The screenshot shows a Java code editor interface with the following components:

- Person.java**: The file name is displayed at the top left.
- Run**: A blue button at the top center.
- Output**: A tab at the top right.
- Code Area**: The main area contains the following Java code:

```
1- class Person {
2     String name;
3-     Person(String name) {
4         this.name = name; }
5-     public static void main(String[] args) {
6         Student s = new Student("Likhitha", 101);
7         s.show(); }}
```
- Output Area**: The right side shows the execution results:

```
Name: Likhitha
Roll No: 101
== Code Execution Successful ==
```