**OBJECT ORIENTED PROGRAMMING**

**LAB# 07 TASKS**

## **In-Depth Understanding the Concept Of Inheritance**

1. Write the classes below containing the given instance variables and methods, following the inherited hierarchy:



1. Write a program that inherits a class named Alien and Pirates from a parent class Human. The human class has its own features like, Human can sleep, walk, talk etc. the Alien and Pirates class inheriting these functionalities as well as they have their characteristics, thus explaining the concepts of inheritance.
2. Write a program that inherits a class named Produce, Cosmetics, Pharmacy, electronic Item and Cloth from a parent class Item. The Item class has its own features like, name and price etc. the Child classes inheriting these functionalities as well as they have their characteristics, thus explaining the concepts of inheritance. Chile classes like Produce, can have their own child classes i.e., Frozen and Fresh.

4) Write a program that inherits a class named Pakistani, BBQ, Chines, Fast Food and Beverages etc. from a parent class Cuisines. The Cuisines class has its own features like, name, quantity and price etc. the Child classes inheriting these functionalities as well as they have their characteristics, thus explaining the concepts of inheritance. Child classes can have their own child classes.

1. Write code according to given guide.You must draw a class diagram first to start writing your code.

Consider a superclass Items which models customer’s purchases. This class has:

* Two private instance variables name (String) and unitPrice (double).
* One constructor to initialize the instance variables.
* A default constructor to initialize name to “no item”, and unitPrice to 0. use this()
* A method getPrice that returns the unitPrice.
* Accessor and mutator methods.
* A toString method to return the name of the item followed by @ symbol, then the unitPrice.

Consider two subclasses WeighedItem and CountedItem. WeighedItem has an additional instance variable weight (double) in Kg while CountedItem has an additional variable quantity (int) both private.

* Write an appropriate constructor for each of the classes making use of the constructor of the superclass in defining those of the subclasses.
* Override getPrice method that returns the price of the purchasedItem based on its unit price and weight (WeighedItem), or quantity (CountedItem). Make use of getPrice of the superclass
* Override also toString method for each class making use of the toString method of the superclass in defining those of the subclasses.

toString should return something that can be printed on the receipt.

For example

Banana @ 3.00 1.37 Kg 4.11 PKR (in case of WeighedItem class)

Pens @ 4.5 10 units 45 PKR (in case of CountedItem class)

**Constructors Example of Inheritance**

**Constructors with Inheritance**

class Shape {

 private int length;

 private int breadth;

 public int getBreadth() {

 return breadth;

 }

 public int getLength() {

 return length;

 }

 public void setBreadth(int i) {

 breadth = i;

 }

 public void setLength(int i) {

 length = i;

 }

 // default Constructor

 Shape() {

 length = 0;

 breadth = 0;

 System.out.println("Inside default constructor of Shape ");

 }

 // Parameterized Constructor

 Shape(int len, int bdth) {

 length = len;

 breadth = bdth;

 System.out.println("Inside constructor of Shape ");

 System.out.println("length : " + length);

 System.out.println("breadth : " + breadth);

 }

}

// A subclass which extends for shape

class Rectangle extends Shape {

 private String type;

 // default Constructor

 Rectangle() {

 super();

 type = null;

 System.out.println("Inside default constructor of rectangle ");

 }

 // Parameterized Constructor

 Rectangle(String ty, int len, int bdth) {

 super (len, bdth);

 System.out.println("Inside constructor of rectangle ");

 System.out.println("length : " + len);

 System.out.println("breadth : " + bdth);

 System.out.println("type : " + type);

 }

 public String getType() {

 return type;

 }

 public void setType(String string) {

 type = string;

 }

}

// A subclass which extends for rectangle

class ColoredRectangle extends Rectangle {

 private String color;

 /\* default Constructor\*/

 ColoredRectangle() {

 super();

 color = null;

 System.out.println("Inside default constructor of coloredRectangle");

 }

 // Parameterized Constructor

 ColoredRectangle(String c, String ty, int len, int bdth) {

 super (ty, len, bdth);

 System.out.println("Inside constructor of coloredRectangle ");

 System.out.println("length : " + len);

 System.out.println("breadth : " + bdth);

 System.out.println("type : " + ty);

 }

 public String getColor() {

 return color;

 }

 public void setColor(String string) {

 color = string;

 }

}

public class Test {

 public static void main(String args[]) {

 ColoredRectangle CR = new ColoredRectangle();

 ColoredRectangle CR2 = new ColoredRectangle("Red","Big", 5, 2 );

 }

}

Output:

Inside default constructor of Shape

Inside default constructor of rectangle

Inside default constructor of coloredRectangle

Inside constructor of Shape

length : 5

breadth : 2

Inside constructor of rectangle

length : 5

breadth : 2

type : null

Inside constructor of coloredRectangle

length : 5

breadth : 2

type : Big