**WEEK 6 – LAB PROBLEM**

Q1. Fruit and Apple Classes

Soln.

class Fruit {

protected String color;

protected String taste;

Fruit(String color, String taste) {

this.color = color;

this.taste = taste;

}

}

class Apple extends Fruit {

private String variety;

Apple(String color, String taste, String variety) {

super(color, taste);

this.variety = variety;

}

void showDetails() {

System.out.println("Apple Details:");

System.out.println("Color: " + color);

System.out.println("Taste: " + taste);

System.out.println("Variety: " + variety);

}

}

public class Main {

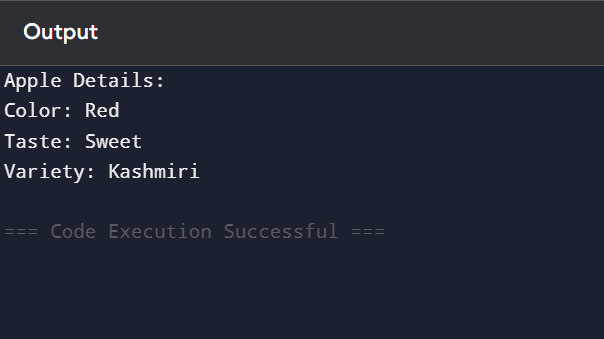
public static void main(String[] args) {

Apple a1 = new Apple("Red", "Sweet", "Kashmiri");

a1.showDetails();

}

}



Q2. Phone and Smart Phone Constructors

Soln.

class Phone {

protected String brand;

protected String model;

Phone() {

System.out.println("Phone default constructor called");

}

Phone(String brand, String model) {

this.brand = brand;

this.model = model;

System.out.println("Phone parameterized constructor called");

}

}

class SmartPhone extends Phone {

private String operatingSystem;

SmartPhone() {

super();

System.out.println("SmartPhone default constructor called");

}

SmartPhone(String brand, String model, String operatingSystem) {

super(brand, model);

this.operatingSystem = operatingSystem;

System.out.println("SmartPhone parameterized constructor called");

}

void showDetails() {

System.out.println("Brand: " + brand);

System.out.println("Model: " + model);

System.out.println("Operating System: " + operatingSystem);

}

}

public class Main {

public static void main(String[] args) {

System.out.println("Creating Phone object:");

Phone p1 = new Phone("Nokia", "1100");

System.out.println("\nCreating SmartPhone object with default constructor:");

SmartPhone sp1 = new SmartPhone();

System.out.println("\nCreating SmartPhone object with parameterized constructor:");

SmartPhone sp2 = new SmartPhone("Apple", "iPhone 15", "iOS");

sp2.showDetails();

}

}



Q3. Bird Flying Behavior

Soln.

class Bird {

void fly() {

System.out.println("A bird can fly in its own way.");

}

}

class Penguin extends Bird {

@Override

void fly() {

System.out.println("Penguins cannot fly, they swim instead.");

}

}

class Eagle extends Bird {

@Override

void fly() {

System.out.println("Eagles fly high and strong.");

}

}

public class Main {

public static void main(String[] args) {

Bird[] birds = new Bird[3];

birds[0] = new Bird();

birds[1] = new Penguin();

birds[2] = new Eagle();

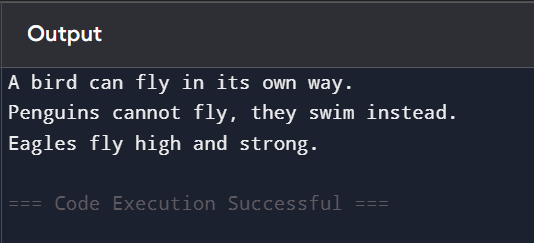
for (Bird b : birds) {

b.fly();

}

}

}



Q4. Color Hierarchy Chain

Soln.

class Color {

protected String name;

Color(String name) {

this.name = name;

System.out.println("Color constructor called");

}

void showColor() {

System.out.println("Color: " + name);

}

}

class PrimaryColor extends Color {

protected String intensity;

PrimaryColor(String name, String intensity) {

super(name);

this.intensity = intensity;

System.out.println("PrimaryColor constructor called");

}

void showPrimaryColor() {

showColor();

System.out.println("Intensity: " + intensity);

}

}

class RedColor extends PrimaryColor {

private String shade;

RedColor(String name, String intensity, String shade) {

super(name, intensity);

this.shade = shade;

System.out.println("RedColor constructor called");

}

void showRedColor() {

showPrimaryColor();

System.out.println("Shade: " + shade);

}

}

public class Main {

public static void main(String[] args) {

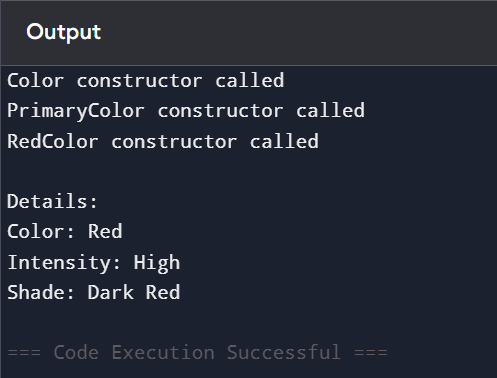
RedColor red = new RedColor("Red", "High", "Dark Red");

System.out.println("\nDetails:");

red.showRedColor();

}

}



Q5. Musical Instrument Family

Soln.

class Instrument {

protected String name;

protected String material;

Instrument(String name, String material) {

this.name = name;

this.material = material;

}

void play() {

System.out.println("Playing a musical instrument.");

}

void showDetails() {

System.out.println("Instrument: " + name);

System.out.println("Material: " + material);

}

}

class Piano extends Instrument {

private int keys;

Piano(String name, String material, int keys) {

super(name, material);

this.keys = keys;

}

@Override

void play() {

System.out.println("Playing piano melodies.");

}

@Override

void showDetails() {

super.showDetails();

System.out.println("Keys: " + keys);

}

}

class Guitar extends Instrument {

private int strings;

Guitar(String name, String material, int strings) {

super(name, material);

this.strings = strings;

}

@Override

void play() {

System.out.println("Strumming guitar chords.");

}

@Override

void showDetails() {

super.showDetails();

System.out.println("Strings: " + strings);

}

}

class Drum extends Instrument {

private String type;

Drum(String name, String material, String type) {

super(name, material);

this.type = type;

}

@Override

void play() {

System.out.println("Beating the drum rhythm.");

}

@Override

void showDetails() {

super.showDetails();

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

}

}

public class Main {

public static void main(String[] args) {

Instrument[] instruments = new Instrument[3];

instruments[0] = new Piano("Grand Piano", "Wood", 88);

instruments[1] = new Guitar("Acoustic Guitar", "Wood", 6);

instruments[2] = new Drum("Bass Drum", "Metal", "Percussion");

for (Instrument inst : instruments) {

inst.play();

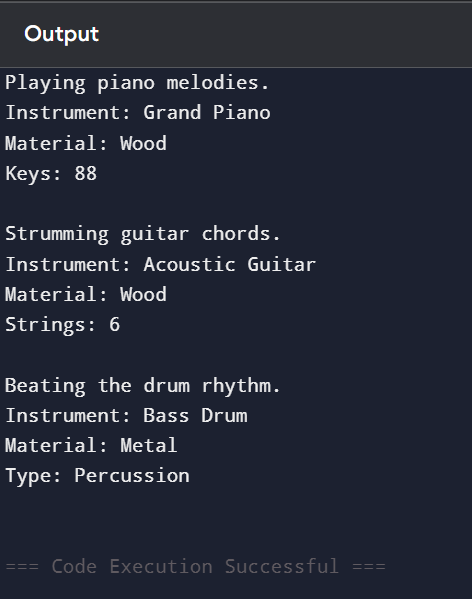
inst.showDetails();

System.out.println();

}

}

}



Q6. Box and Gift Box Enhancement

Soln.

class Box {

void pack() {

System.out.println("Packing a box.");

}

void unpack() {

System.out.println("Unpacking a box.");

}

}

class GiftBox extends Box {

@Override

void pack() {

super.pack();

System.out.println("Adding wrapping paper and ribbon for the gift.");

}

@Override

void unpack() {

super.unpack();

System.out.println("Removing wrapping paper and ribbon from the gift.");

}

}

public class Main {

public static void main(String[] args) {

System.out.println("Normal Box:");

Box b = new Box();

b.pack();

b.unpack();

System.out.println("\nGift Box:");

GiftBox gb = new GiftBox();

gb.pack();

gb.unpack();

}

}

