### **1. Stack and Heap**

public class MemoryExample {

public static void main(String[] args) {

// Stack example

int x = 10; // x is stored in the stack

// Heap example

Integer y = new Integer(20); // Integer object is stored in the heap

}

}

Explanation:

* **Stack**: x is a primitive int and its value is directly stored in the stack memory.
* **Heap**: y is an Integer object, which is created using the new keyword and stored in the heap memory.

### 

### **2. Class and Constructor**

public class Car {

String color;

int year;

// Constructor

public Car(String c, int y) {

color = c;

year = y;

}

public static void main(String[] args) {

// Creating an object of Car class

Car myCar = new Car("Red", 2023);

System.out.println("My car is " + myCar.color + " and was made in " + myCar.year);

}

}

Explanation:

* **Class**: Car defines the blueprint for creating car objects with color and year attributes.
* **Constructor**: Car(String c, int y) is a constructor that initializes a Car object with the provided color and year values when new Car("Red", 2023) is called.

### 

### **3. Constructor Overloading**

public class Rectangle {

int width, height;

// First constructor with two parameters

public Rectangle(int w, int h) {

width = w;

height = h;

}

// Second constructor with one parameter (overloaded)

public Rectangle(int sideLength) {

width = sideLength;

height = sideLength;

}

public static void main(String[] args) {

Rectangle rect1 = new Rectangle(5, 10);

Rectangle square = new Rectangle(7);

System.out.println("Rectangle dimensions: " + rect1.width + "x" + rect1.height);

System.out.println("Square dimensions: " + square.width + "x" + square.height);

}

}

Explanation:

* **Constructor Overloading**: Rectangle class has two constructors - one that initializes a rectangle with specified width and height, and another that initializes a square with a single side length.

### 

### **4. Copy Constructor**

public class Student {

String name;

int age;

// Constructor to initialize name and age

public Student(String n, int a) {

name = n;

age = a;

}

// Copy constructor

public Student(Student original) {

name = original.name;

age = original.age;

}

public static void main(String[] args) {

Student student1 = new Student("Alice", 20);

Student student2 = new Student(student1); // Using copy constructor

System.out.println("Student 1: " + student1.name + " (" + student1.age + " years old)");

System.out.println("Student 2: " + student2.name + " (" + student2.age + " years old)");

}

}

Explanation:

* **Copy Constructor**: Student(Student original) is a constructor that takes another Student object as an argument and creates a new Student object with the same name and age.

### 

### **5. Object Creation**

public class Dog {

String breed;

int age;

// Constructor to initialize breed and age

public Dog(String b, int a) {

breed = b;

age = a;

}

public static void main(String[] args) {

// Creating objects of Dog class

Dog dog1 = new Dog("Labrador", 3);

Dog dog2 = new Dog("Poodle", 5);

// Accessing object properties

System.out.println("Dog 1 is a " + dog1.breed + " and is " + dog1.age + " years old.");

System.out.println("Dog 2 is a " + dog2.breed + " and is " + dog2.age + " years old.");

}

}

Explanation:

* **Object Creation**: Dog class demonstrates how to create multiple instances (dog1 and dog2) of the class using the new keyword, each initialized with specific breed and age.

Prepared by : Rishi Patel