1. class Parent {

void even() {

System.out.println("Even numbers from 1 to 20:");

for (int i = 2; i <= 20; i += 2) {

System.out.print(i + " ");

}

System.out.println();

}

void odd() {

System.out.println("Odd numbers from 1 to 20:");

for (int i = 1; i <= 20; i += 2) {

System.out.print(i + " ");

}

System.out.println();

}

void prime() {

System.out.println("Prime numbers from 1 to 20:");

for (int i = 2; i <= 20; i++) {

boolean isPrime = true;

for (int j = 2; j <= i / 2; j++) {

if (i % j == 0) {

isPrime = false;

break;

}

}

if (isPrime) {

System.out.print(i + " ");

}

}

System.out.println();

}

}

class Child extends Parent {

@Override

void even() {

System.out.println("Even numbers from 1 to 30:");

for (int i = 2; i <= 30; i += 2) {

System.out.print(i + " ");

}

System.out.println();

}

@Override

void odd() {

System.out.println("Odd numbers from 1 to 30:");

for (int i = 1; i <= 30; i += 2) {

System.out.print(i + " ");

}

System.out.println();

}

@Override

void prime() {

System.out.println("Prime numbers from 1 to 30:");

for (int i = 2; i <= 30; i++) {

boolean isPrime = true;

for (int j = 2; j <= i / 2; j++) {

if (i % j == 0) {

isPrime = false;

break;

}

}

if (isPrime) {

System.out.print(i + " ");

}

}

System.out.println();

}

}

public class Main {

public static void main(String[] args) {

Child child = new Child();

child.even();

child.odd();

child.prime();

child.disp();

}

}

2.class Parent {

int a;

int b;

// Default constructor

Parent() {

this.a = 10;

this.b = 15;

}

// Parameterized constructor

Parent(int a, int b) {

this.a = a;

this.b = b;

}

void disp() {

System.out.println("a = " + a + ", b = " + b);

}

}

public class Main {

public static void main(String[] args) {

Parent obj1 = new Parent();

obj1.disp(); // Output: a = 10, b = 15

Parent obj2 = new Parent(20, 25);

obj2.disp(); // Output: a = 20, b = 25

}

}

3. class Base {

void area(int a, int b) {

System.out.println("Area of square: " + (a \* a));

}

}

class Derived extends Base {

@Override

void area(int a, int b) {

System.out.println("Area of rectangle: " + (a \* b));

}

}

public class Main {

public static void main(String[] args) {

Base baseObj = new Base();

baseObj.area(5, 5); // Output: Area of square: 25

Derived derivedObj = new Derived();

derivedObj.area(5, 10); // Output: Area of rectangle: 50

}

}

4.A. class MathOperations {

int sum(int a, int b) {

return a + b;

}

int sum(int a, int b, int c) {

return a + b + c;

}

}

public class Main {

public static void main(String[] args) {

MathOperations math = new MathOperations();

System.out.println("Sum of two numbers: " + math.sum(5, 10)); // Output: Sum of two numbers: 15

System.out.println("Sum of three numbers: " + math.sum(5, 10, 15)); // Output: Sum of three numbers: 30

}

}

B. class MathOperations {

int sum(int a, int b) {

return a + b;

}

double sum(double a, double b) {

return a + b;

}

}

public class Main {

public static void main(String[] args) {

MathOperations math = new MathOperations();

System.out.println("Sum of two integers: " + math.sum(5, 10)); // Output: Sum of two integers: 15

System.out.println("Sum of two doubles: " + math.sum(5.5, 10.5)); // Output: Sum of two doubles: 16.0

}

}

C. class MathOperations {

int sum(int a, int b) {

return a + b;

}

int sum(int b, int a, int c) {

return a + b + c;

}

}

public class Main {

public static void main(String[] args) {

MathOperations math = new MathOperations();

System.out.println("Sum of two numbers: " + math.sum(5, 10)); // Output: Sum of two numbers: 15

System.out.println("Sum of three numbers (interchanged parameters): " + math.sum(5, 10, 15)); // Output: Sum of three numbers (interchanged parameters): 30

}

}

5. abstract class Saveetha {

abstract void total();

abstract void average();

void mean() {

System.out.println("Mean method implementation");

}

void mode() {

System.out.println("Mode method implementation");

}

void median() {

System.out.println("Median method implementation");

}

}

class ConcreteClass extends Saveetha {

@Override

void total() {

System.out.println("Total method implementation");

}

@Override

void average() {

System.out.println("Average method implementation");

}

}

public class Main {

public static void main(String[] args) {

ConcreteClass obj = new ConcreteClass();

obj.total();

obj.average();

obj.mean();

obj.mode();

obj.median();

}

}

6. interface Operations {

void add();

void sub();

void mul();

void div();

}

class Calculator implements Operations {

@Override

public void add() {

System.out.println("Add method implementation");

}

@Override

public void sub() {

System.out.println("Subtract method implementation");

}

@Override

public void mul() {

System.out.println("Multiply method implementation");

}

@Override

public void div() {

System.out.println("Divide method implementation");

}

}

public class Main {

public static void main(String[] args) {

Calculator calc = new Calculator();

calc.add();

calc.sub();

calc.mul();

calc.div();

}

}

7. interface Sum {

void sum();

}

interface Avg {

void avg();

}

interface Percentage {

void percentage();

}

class Statistics implements Sum, Avg, Percentage {

@Override

public void sum() {

System.out.println("Sum method implementation");

}

@Override

public void avg() {

System.out.println("Average method implementation");

}

@Override

public void percentage() {

System.out.println("Percentage method implementation");

}

}

public class Main {

public static void main(String[] args) {

Statistics stats = new Statistics();

stats.sum();

stats.avg();

stats.percentage();

}

}

8. // Define the Tree interface

interface Tree {

void fruits(String fruit1, String fruit2);

void leaves(String leaf1, String leaf2, String leaf3);

void flowers(String flower1, String flower2, String flower3, String flower4);

}

// Implement Branch1 class

class Branch1 implements Tree {

@Override

public void fruits(String fruit1, String fruit2) {

System.out.println("Fruits on Branch1: " + fruit1 + ", " + fruit2);

}

@Override

public void leaves(String leaf1, String leaf2, String leaf3) {

System.out.println("Leaves on Branch1: " + leaf1 + ", " + leaf2 + ", " + leaf3);

}

@Override

public void flowers(String flower1, String flower2, String flower3, String flower4) {

System.out.println("Flowers on Branch1: " + flower1 + ", " + flower2 + ", " + flower3 + ", " + flower4);

}

}

// Implement Branch2 class

class Branch2 implements Tree {

@Override

public void fruits(String fruit1, String fruit2) {

System.out.println("Fruits on Branch2: " + fruit1 + ", " + fruit2);

}

@Override

public void leaves(String leaf1, String leaf2, String leaf3) {

System.out.println("Leaves on Branch2: " + leaf1 + ", " + leaf2 + ", " + leaf3);

}

@Override

public void flowers(String flower1, String flower2, String flower3, String flower4) {

System.out.println("Flowers on Branch2: " + flower1 + ", " + flower2 + ", " + flower3 + ", " + flower4);

}

}

public class Main {

public static void main(String[] args) {

// Creating objects of Branch1 and Branch2

Branch1 branch1 = new Branch1();

Branch2 branch2 = new Branch2();

// Calling methods on Branch1

branch1.fruits("Apple", "Orange");

branch1.leaves("Green", "Yellow", "Brown");

branch1.flowers("Rose", "Lily", "Sunflower", "Tulip");

// Calling methods on Branch2

branch2.fruits("Mango", "Banana");

branch2.leaves("Dark Green", "Light Green", "Golden Yellow");

branch2.flowers("Hibiscus", "Daisy", "Orchid", "Carnation");

}

}

9. A. class MyClass {

static int staticVariable = 10;

public static void main(String[] args) {

System.out.println("Static variable: " + MyClass.staticVariable);

}

}

B. class MyClass {

static void staticMethod() {

System.out.println("This is a static method.");

}

public static void main(String[] args) {

MyClass.staticMethod();

}

}

C . class MyClass {

static {

System.out.println("This is a static block.");

}

public static void main(String[] args) {

// No need to call the static block, it will be executed automatically when the class is loaded.

}

}

D. class OuterClass {

static class NestedStaticClass {

void display() {

System.out.println("This is a static nested class.");

}

}

public static void main(String[] args) {

OuterClass.NestedStaticClass nested = new OuterClass.NestedStaticClass();

nested.display();

}

}

10. A. class MyClass {

final int finalVariable = 10;

public static void main(String[] args) {

MyClass obj = new MyClass();

System.out.println("Final variable: " + obj.finalVariable);

}

}

B. class Parent {

final void finalMethod() {

System.out.println("This is a final method.");

}

}

class Child extends Parent {

public static void main(String[] args) {

Child obj = new Child();

obj.finalMethod();

}

}

C. final class FinalClass {

void display() {

System.out.println("This is a final class.");

}

public static void main(String[] args) {

FinalClass obj = new FinalClass();

obj.display();

}

}

11.A.class MyClass {

int num;

MyClass(int num) {

this.num = num; // 'this' is used to refer to the instance variable

}

public static void main(String[] args) {

MyClass obj = new MyClass(10);

System.out.println("Value of num: " + obj.num);

}

}

B. class MyClass {

int num;

MyClass() {

this(100); // 'this' is used to invoke another constructor of the same class

}

MyClass(int num) {

this.num = num;

}

public static void main(String[] args) {

MyClass obj = new MyClass();

System.out.println("Value of num: " + obj.num);

}

}

C. class MyClass {

void display() {

System.out.println("Inside display method.");

}

void invokeDisplay() {

this.display(); // 'this' is used to pass the current object as a parameter

}

public static void main(String[] args) {

MyClass obj = new MyClass();

obj.invokeDisplay();

}

}

D. class MyClass {

int num;

MyClass(int num) {

this.num = num;

}

MyClass getObject() {

return this; // 'this' is used to return the current object

}

public static void main(String[] args) {

MyClass obj1 = new MyClass(10);

MyClass obj2 = obj1.getObject();

System.out.println("Value of num: " + obj2.num);

}

}

12.A.class Parent {

int num = 100;

}

class Child extends Parent {

void display() {

System.out.println("Value of num in parent class: " + super.num); // 'super' is used to access superclass members

}

public static void main(String[] args) {

Child obj = new Child();

obj.display();

}

}

B. class Parent {

Parent() {

System.out.println("This is parent class constructor.");

}

}

class Child extends Parent {

Child() {

super(); // 'super' is used to call superclass constructor

System.out.println("This is child class constructor.");

}

public static void main(String[] args) {

Child obj = new Child();

}

}

C. class Parent {

void display() {

System.out.println("This is parent class method.");

}

}

class Child extends Parent {

void display() {

super.display(); // 'super' is used to invoke superclass methods

System.out.println("This is child class method.");

}

public static void main(String[] args) {

Child obj = new Child();

obj.display();

}

}

13. public class Box<T> {

private T value;

public void setValue(T value) {

this.value = value;

}

public T getValue() {

return value;

}

}

14.public class Pair<K, V> {

private K key;

private V value;

public Pair(K key, V value) {

this.key = key;

this.value = value;

}

public K getKey() {

return key;

}

public V getValue() {

return value;

}

}

15.public class Util {

public static <T> T doSomething(T input) {

// Do something with input

return input;

}

}

16.public class NumberBox<T extends Number> {

private T value;

public void setValue(T value) {

this.value = value;

}

public T getValue() {

return value;

}

}

17.import java.util.Scanner;

public class InputExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a character: ");

char ch = scanner.next().charAt(0);

System.out.print("Enter a string: ");

String str = scanner.next();

System.out.print("Enter an integer: ");

int numInt = scanner.nextInt();

System.out.print("Enter a float: ");

float numFloat = scanner.nextFloat();

System.out.print("Enter a double: ");

double numDouble = scanner.nextDouble();

System.out.println("Character: " + ch);

System.out.println("String: " + str);

System.out.println("Integer: " + numInt);

System.out.println("Float: " + numFloat);

System.out.println("Double: " + numDouble);

scanner.close();

}

}

18.import java.util.Date;

public class DateTimeExample {

public static void main(String[] args) {

Date currentDate = new Date();

System.out.println("Current Date and Time: " + currentDate);

}

}

19.import java.util.UUID;

public class UUIDExample {

public static void main(String[] args) {

UUID uuid = UUID.randomUUID();

System.out.println("Random UUID: " + uuid);

}

}

20.public class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

@Override

public String toString() {

return "Person{name='" + name + "', age=" + age + "}";

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null || getClass() != obj.getClass())

return false;

Person person = (Person) obj;

return age == person.age && name.equals(person.name);

}

public static void main(String[] args) {

Person person1 = new Person("John", 30);

Person person2 = new Person("John", 30);

System.out.println("toString() Example: " + person1.toString());

System.out.println("equals() Example: " + person1.equals(person2));

}

}