**Program: -6**

**Objective**: Write a Program in java to illustrate the concept of local, instance and static variable.

**Code:**

*public* *class* variables {

*static* int staticVar = 10;

    int instanceVar;

*public* variables(int instanceVar) {

        this.instanceVar = instanceVar;

    }

*public* void demonstrateLocalVariable() {

        int localVar = 5;

        System.out.println("Local variable: " + localVar);

    }

*public* *static* void main(String[] args) {

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

        variables example = new variables(20);

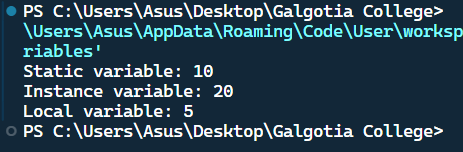
        System.out.println("Instance variable: " + example.instanceVar);

        example.demonstrateLocalVariable();

    }

}

**Output:**



**Program: -7**

**Objective**: WAP in java to implement implicit and explicit type casting.

**Code:**

*public* *class* typeCasting {

*public* *static* void main(String[] args) {

*// Implicit type casting*

        int i = 10;

        long l = i;

*// Explicit type casting*

        double d = 10.5;

        int j = (int) d;

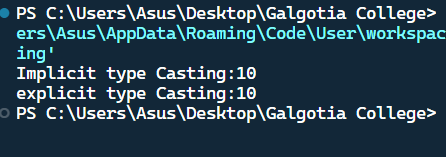
        System.out.println("Implicit type Casting:" + l);

        System.out.println("explicit type Casting:"+ j);

    }

}

**Output:**

****

**Program: -8**

**Objective**: WAP in java for implement the various operators in java;

**Code:**

*public* *class* Operators {

*public* *static* void main(String[] args) {

        int a = 10, b = 5;

*// Arithmetic Operators*

        System.out.println("Arithmetic Operators:");

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

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

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

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

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

*// Relational Operators*

        System.out.println("\nRelational Operators:");

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

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

        System.out.println("a > b: " + (a > b));

        System.out.println("a < b: " + (a < b));

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

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

*// Logical Operators*

        System.out.println("\nLogical Operators:");

        System.out.println("(a > b) && (a < 15): " + ((a > b) && (a < 15)));

        System.out.println("(a > b) || (b < 2): " + ((a > b) || (b < 2)));

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

*// Bitwise Operators*

        System.out.println("\nBitwise Operators:");

        System.out.println("a & b: " + (a & b));

        System.out.println("a | b: " + (a | b));

        System.out.println("a ^ b: " + (a ^ b));

        System.out.println("~a: " + (~a));

        System.out.println("a << 2: " + (a << 2));

        System.out.println("a >> 2: " + (a >> 2));

*// Assignment Operators*

        System.out.println("\nAssignment Operators:");

        a += b;

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

        a -= b;

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

        a \*= b;

        System.out.println("a \*= b: " + a);

        a /= b;

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

        a %= b;

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

*// Unary Operators*

        System.out.println("\nUnary Operators:");

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

        System.out.println("--a: " + (--a));

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

        System.out.println("a--: " + (a--));

*// Ternary Operator*

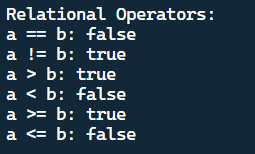
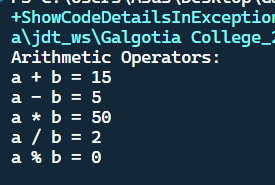
        System.out.println("\nTernary Operator:");

        System.out.println("a > b ? a : b: " + (a > b ? a : b));

    }

}

**Output:**



**Program: -9**

**Objective**: WAP in java for constructor overloading.

**Code:**

*public* *class* ConstructorOverloadingExample {

    int id;

    String name;

    int age;

*public* ConstructorOverloadingExample() {

        this.id = 0;

        this.name = "Unknown";

        this.age = 0;

    }

*public* ConstructorOverloadingExample(int id) {

        this.id = id;

        this.name = "Unknown";

        this.age = 0;

    }

*public* ConstructorOverloadingExample(int id, String name) {

        this.id = id;

        this.name = name;

        this.age = 0;

    }

*public* ConstructorOverloadingExample(int id, String name, int age) {

        this.id = id;

        this.name = name;

        this.age = age;

    }

*public* void display() {

        System.out.println("ID: " + id + ", Name: " + name + ", Age: " + age);

    }

*public* *static* void main(String[] args) {

*// Creating objects using different constructors*

        ConstructorOverloadingExample obj1 = new ConstructorOverloadingExample();

        ConstructorOverloadingExample obj2 = new ConstructorOverloadingExample(1);

        ConstructorOverloadingExample obj3 = new ConstructorOverloadingExample(2, "Alice");

        ConstructorOverloadingExample obj4 = new ConstructorOverloadingExample(3, "Bob", 25);

      obj1.display();

        obj2.display();

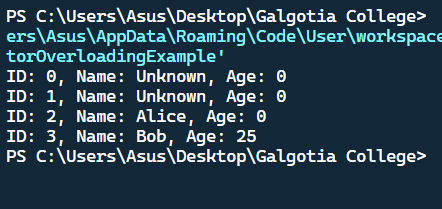
        obj3.display();

        obj4.display();

    }

}

**Output:**



**Program: -10**

**Objective**: WAP in java for method Overloading.

**Code:**

*public* *class* methodOverloading {

*public* void display() {

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

    }

*public* void display(int a) {

        System.out.println("the value is :" + a);

    }

*public* void display(String name) {

        System.out.println("the name is :" + name);

    }

*public* *static* void main(String[] args) {

        methodOverloading T = new methodOverloading();

        T.display();

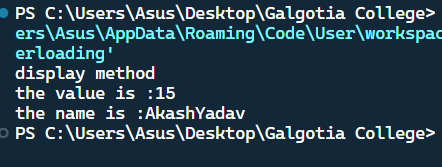
        T.display(15);

        T.display("AkashYadav");

    }

}

**Output:**

****