1. Write a Java program to demonstrate method overloading with two methods having different data types.

A.

```
public class MethodOverloadingExample {
   public int add(int a, int b) {
      return a + b;
   }
   public double add(double a, double b) {
      return a + b;
   }
   public static void main(String[] args) {
      MethodOverloadingExample example = new MethodOverloadingExample();
      int sumInt = example.add(10, 20);
      System.out.println("Sum of integers: " + sumInt);
      double sumDouble = example.add(10.5, 20.5);
      System.out.println("Sum of doubles: " + sumDouble);
   }
}
```

2. Create a Java program to demonstrate method overloading withtwo methods having different numbers of parameters.

A.

```
public class MethodOverloadingExample {

   public int multiply(int a, int b) {

      return a * b;

   }

   public int multiply(int a, int b, int c) {

      return a * b * c;
```

```
    }

    public static void main(String[] args) {

        MethodOverloadingExample example = new MethodOverloadingExample();

        int productTwo = example.multiply(10, 20);

        System.out.println}
```

3. Write a Java program to demonstrate method overloading with two methods having different data types and a common parameter?

A.

```
public class MethodOverloadingExample {

    public int calculateArea(int side) {

        return side * side;

    }

    public double calculateArea(double side) {

        return side * side;

    }

    public static void main(String[] args) {

        MethodOverloadingExample example = new MethodOverloadingExample();

        int areaInt = example.calculateArea(5);

        System.out.println("Area of square with integer side: " + areaInt);

        double areaDouble = example.calculateArea(5.5);
        System.out.println("Area of square with double side: " + areaDouble);

    }

}
```

4. Create a Java program to demonstrate method overloading with amixture of data types and parameter counts?

A.

```
public class MethodOverloadingExample {

    // Method to calculate the volume of a cube (int side)

    public int calculateVolume(int side) {

        return side * side * side;

    }

    public int calculateVolume(int length, int width, int height) {

        return length * width * height;

    }

    public double calculateVolume(double radius, int height) {

        return Math.PI * radius * radius * height;

    }

    public static void main(String[] args) {

        MethodOverloadingExample example = new MethodOverloadingExample();

        int volumeCube = example.calculateVolume(5);
        System.out.println("Volume of cube: " + volumeCube);

        int volumeCuboid = example.calculateVolume(2, 3, 4);

        System.out.println("Volume of cuboid: " + volumeCuboid);

        double volumeCylinder = example.calculateVolume(2.5, 5);

        System.out.println("Volume of cylinder: " + volumeCylinder);

    }

}
```

5.Write a Java program to demonstrate method overloading withchanging the number of parameters and data types?

A.

```
public class MethodOverloadingExample {

    public int calculateArea(int side) {

        return side * side;

    }   public int calculateArea(int length, int width) {

        return length * width;

    }

    public double calculateArea(double radius) {

        return Math.PI * radius * radius;

    }

    public double calculateArea(double base, double height) {
        return 0.5 * base * height;

    }

    public static void main(String[] args) {

        MethodOverloadingExample example = new MethodOverloadingExample();

        int areaSquare = example.calculateArea(5);

        System.out.println("Area of square: " + areaSquare);

        int areaRectangle = example.calculateArea(2, 3);

        System.out.println("Area of rectangle: " + areaRectangle);

        double areaCircle = example.calculateArea(2.5);

        System.out.println("Area of circle: " + areaCircle);

        double areaTriangle = example.calculateArea(4.0, 3.0);

        System.out.println("Area of triangle: " + areaTriangle);

    }

}
```

6. Create a Java program to demonstrate method overloading withoverloaded methods that use different parameter types and returntypes, including type casting.

A.

```
public class MethodOverloadingExample {

    public int add(int a, int b) {

        return a + b;
```

```
}

public double add(double a, double b) {

    return a + b;

}

public double add(int a, double b) {

    // Type casting 'a' to double before adding

    return (double) a + b;

}

public double add(double a, int b) {

    // Type casting 'b' to double before adding

    return a + (double) b;

}

public static void main(String[] args) {

    MethodOverloadingExample example = new MethodOverloadingExample();

    int sumInt = example.add(10, 20);

    System.out.println("Sum of integers: " + sumInt);

    double sumDouble = example.add(10.5, 20.5);

    System.out.println("Sum of doubles: " + sumDouble);

    double sumIntDouble = example.add(10, 20.5);

    System.out.println("Sum of integer and double: " + sumIntDouble);
```

```
        double sumDoubleInt = example.add(10.5, 20);

        System.out.println("Sum of double and integer: " + sumDoubleInt);

    }

}
```

7. Write a Java program to demonstrate method overloading withoverloaded methods that use different parameter types, including booleans and boolean arrays?

A.

```
public class MethodOverloadingExample {

    public void print(boolean value) {

        System.out.println("Boolean value: " + value);

    }

    public void print(boolean[] values) {

        System.out.print("Boolean array: ");

        for (boolean value : values) {

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

        }

        System.out.println();

    }

    public void print(boolean value1, boolean value2) {

        System.out.println("Boolean values: " + value1 + ", " + value2);
    }

    public static void main(String[] args) {

        MethodOverloadingExample example = new MethodOverloadingExample();

        example.print(true);

        boolean[] array = {true, false, true};

        example.print(array);

        example.print(true, false);

    }

}
```

8. Create a Java program to demonstrate method overriding with asubclass that overrides a method with a different return type?

A.

```
class Animal {

    public int getLegs() {

        return 4;

    }

}

class Dog extends Animal {

    public double getLegs() {

        return 4.0; // Dog has 4 legs, returning as double
    }
}
public class MethodOverridingExample {
    public static void main(String[] args) {
        Animal animal = new Animal();
        System.out.println("Animal legs: " + animal.getLegs()); // Output: Animal legs: 4
        Dog dog = new Dog();
        System.out.println("Dog legs: " + dog.getLegs()); // Output: Dog legs: 4.0
    }
}
```

9. Write a Java program to demonstrate method overriding with a subclass that overrides a method and calls the superclass method using the super keyword?

A.

```
class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    public void makeSound() {
        super.makeSound(); // Calling superclass method
        System.out.println("Dog barks"); // Adding subclass-specific behavior
    }
}
public class MethodOverridingExample {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.makeSound(); // Output: Animal makes a sound
        System.out.println("---");
        Dog dog = new Dog();
        dog.makeSound();
    }
}
```

10. Write a Java program to demonstrate method overriding witha subclass that overrides a method and changes the access modifier from public to private?

A.

```
class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    private void makeSound() {
        System.out.println("Dog barks");
    }
}
public class MethodOverridingExample {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.makeSound(); // Output: Animal makes a sound
        Dog dog = new Dog();
        dog.makeSound();
    }
}
```

11. Create a Java program to demonstrate method overriding with a subclass that overrides a method and adds a new parameterwith a default value.

A.

```
class Animal {
    public void makeSound(String sound) {
        System.out.println("Animal makes " + sound);
    }
}
class Dog extends Animal {
    public void makeSound(String sound, int times) {
        for (int i = 0; i < times; i++) {
            System.out.println("Dog barks: " + sound);
        }
    }
}
public class MethodOverloadingExample {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.makeSound("some noise;
        Dog dog = new Dog();
        dog.makeSound("woof", 3);
    }
}
```