

Q1.

```
class A {  
    public static void main(String args[])  
    {  
        int var1 = 15;  
        int var2 = 5;  
        int var3 = 0;  
        int ans1 = var1 / var2;  
  
        int ans2 = var1 / var3;  
  
        System.out.println(  
            "Division of va1"  
            + " by var2 is: "  
            + ans1);  
        System.out.println(  
            "Division of va1"  
            + " by var3 is: "  
            + ans2);  
    }  
}
```

Q2.

```
public class main {  
    public static void main(String[] args) {  
        int[] numbers = {1, 2, 3, 4, 5};  
        System.out.println(numbers[5]);  
    }  
}
```

Q3.

```
public class DTE {
```

```
public static void main(String[] args) {  
    int num = "100";  
    System.out.println("Number: " + num);  
}  
}
```

Q4.

```
public class IR {  
    public static void main(String[] args) {  
        rF();  
    }  
  
    public static void rF() {  
        rF();  
    }  
}
```

Q5.

```
public class MR {  
    public static void main(String[] args) {  
        System.out.println("Result: " + calculate());  
    }  
  
    public static int calculate() {  
        int x = 10;  
        int y = 20;  
    }  
}
```

Q6.

```
public class MSE {  
    public static void main(String[] args) {  
        int result = add(5, 10);  
    }  
}
```

```
        System.out.println("Result: " + result);
    }

    public static double add(int a, int b) {
        return a + b;
    }
}
```

Q7.

```
class Parent {
    Parent(int value) {
        System.out.println("Parent class constructor called with value: " + value);
    }
}

class Child extends Parent {
    Child() {
        System.out.println("Child class constructor called");
    }
}

public class CE {
    public static void main(String[] args) {
        Child child = new Child();
    }
}
```

Q8.

```
public class SE {
    int instanceVariable = 10;

    public static void main(String[] args) {
        System.out.println("Instance Variable: " + instanceVariable);
    }
}
```

```
}  
}
```

Q9.

```
class Animal {  
    public Animal(String name) {  
        System.out.println("Animal constructor: " + name);  
    }  
}  
  
class Dog extends Animal {  
    public Dog(String name) {  
        String dog_name = name;  
        super(dog_name);  
    }  
}  
  
public class IE {  
    public static void main(String[] args) {  
        Dog dog = new Dog("Tommy");  
    }  
}
```

Q10.

```
public class CE {  
    public static void main(String[] args) {  
        Object obj = new String("Hello");  
        Integer num = (Integer) obj;  
        System.out.println("Number: " + num);  
    }  
}
```

Q11.

```
public class STE {  
    int value = 42;  
  
    public static void main(String[] args) {  
        System.out.println("Value: " + this.value);  
    }  
}
```

Q12.

```
class User {  
    String name;  
  
    public String getName() {  
        return name;  
    }  
}  
  
public class NPE {  
    public static void main(String[] args) {  
        User user = null;  
        System.out.println("User name: " + user.getName());  
    }  
}
```

Q13.

```
class Bike {  
    final void run() {  
        System.out.println("running");  
    }  
}  
  
class Honda extends Bike {  
    void run() {  
        System.out.println("running safely with 100kmph");  
    }  
}
```

```

    }
}

public class fKW {
    public static void main(String args[]) {
        Honda honda = new Honda();
        honda.run();
    }
}

```

Q14.

```

class MyClass {
    MyClass() {
        System.out.println(privateVariable);
    }

    private int privateVariable = 15;

    private int privateMethod(int a, int b) {
        return a + b;
    }

    public void publicMethod() {
        System.out.println(privateMethod(5, 6));
    }
}

class MyChildClass extends MyClass {
    MyChildClass() {
        privateMethod(5, 6);
        System.out.println(MyClass.privateVariable);
    }
}

public class pAM {
    public static void main(String[] args) {

        MyClass obj = new MyClass();
        obj.privateMethod(5, 6);
        System.out.println(obj.privateVariable);

        MyChildClass obj2 = new MyChildClass();
    }
}

```

```
}  
}
```

Q15.

```
class Parent {  
    void show() {  
        System.out.println("Parent's show()");  
    }  
}  
  
class Child extends Parent {  
    @Override  
    void shoe() {  
        System.out.println("Child's show()");  
    }  
}  
  
public class overridingTest {  
    public static void main(String[] args) {  
        Parent obj1 = new Parent();  
        obj1.show();  
  
        Parent obj2 = new Child();  
        obj2.show();  
    }  
}
```

Q16.

```
class Animal {  
    public void speak() {  
        System.out.println("Animal speaks");  
    }  
}  
  
class Dog extends Animal {  
    public void speak(String sound) {  
        System.out.println("Dog barks: " + sound);  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Animal animal = new Dog();  
        animal.speak();  
        animal.speak("Woof");  
    }  
}
```

Q17.

```
class Test {  
    public static void main(String[] args) {  
        int a = 1;  
        do {  
            a=a+1;  
            System.out.println(a);  
        }  
        while(a !=0);  
    }  
}
```

Q18.

```
class Animal {  
    private String name;  
  
    public Animal(String name) {  
        this.name = name;  
    }  
  
    private void printName() {  
        System.out.println("Name: " + name);  
    }  
}
```



```
class Dog extends Animal {  
    public Dog(String name) {  
        super(name);  
    }  
  
    public void showName() {  
        super.printName();  
    }  
}  
  
public class Test {  
    public static void main(String[] args) {  
        Dog d = new Dog("Rex");  
        d.showName();  
    }  
}
```

Q19.

```
class Animal {  
    public final void speak() {  
        System.out.println("Animal speaks");  
    }  
}  
  
class Dog extends Animal {  
    @Override  
    void speak() {  
        System.out.println("Dog Barks");  
    }  
}
```

Q20.

```
class Animal extends Dog {  
    void speak() {  
        System.out.println("Animal speaks");  
    }  
}  
  
class Dog extends Animal {  
    @Override  
    void speak() {  
        System.out.println("Dog Barks");  
    }  
}
```