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

Q11.

```
public class StaticThisError {
   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 NullPointerError {
   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 finalKW {
  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); // Error
    System.out.println(MyClass.privateVariable); // Error
  }
}

public class privateAM {
  public static void main(String[] args) {
    // Code goes here

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

    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.

```
import java.util.Scanner;

class Employee {
  String name, address, job_title;
  double salary;
  int experience;

  Employee(String n, String a, String j, double s, int e) {
    name = n;
    address = a;
    job_title = j;
    salary = s;
    experience = e;
  }

  void calculateBonus() {
```

```
    System.out.println("Bonus of " + name + ": " + 0.1 * salary);
  }

  void generatePerformanceReport(String per_type) {
    System.out.println("Performance Report of " + name + ": " + per_type);
  }

  void manageProjects(int no_of_projects) {
    System.out.println("Projects managed by " + name + ": " + no_of_projects);
  }

  void info() {
    System.out.println("Employee's Name: " + name);
    System.out.println("Employee's Address: " + address);
    System.out.println("Employee's Job Title: " + job_title);
    System.out.println("Employee's Salary: " + salary);
    System.out.println();
  }
}

class Manager extends Employee {

  Manager(String n, String a, String j, double s, int e) {
    super(n, a, j, s, e);
  }
}

class Developer extends Employee {
  Developer(String n, String a, String j, double s, int e) {
    super(n, a, j, s, e);
  }
}

class Programmer extends Employee {
  Programmer(String n, String a, double s, int e) {
    String j = "Programmer";
    super(n, a, j, s, e);
  }
}

public class EmpHierar {
  public static void main(String[] args) {
    // Creating objects
    Scanner sc = new Scanner(System.in);
    Manager mgr;
```

```
Developer dev;
Programmer prog;

// Manager
System.out.print("Enter Manager's Name: ");
String mgr_name = sc.next();

System.out.print("Enter Manager's Address: ");
String mgr_address = sc.next();

System.out.print("Enter Manager's Salary (in $): ");
double mgr_salary = sc.nextDouble();

System.out.print("Enter Manager's Experience (in years): ");
int mgr_exp = sc.nextInt();

System.out.print("Enter Manager's Performance Type (Bad/Good/Excellent): ");
String mgr_per_type = sc.next();

System.out.print("Enter Manager's Number of Projects Managed: ");
int mgr_no_of_projects = sc.nextInt();

System.out.println();

// Developer
System.out.print("Enter Developer's Name: ");
String dev_name = sc.next();

System.out.print("Enter Developer's Address: ");
String dev_address = sc.next();

System.out.print("Enter Developer's Salary (in $): ");
double dev_salary = sc.nextDouble();

System.out.print("Enter Developer's Experience (in years): ");
int dev_exp = sc.nextInt();

System.out.print("Enter Developer's Performance Type (Bad/Good/Excellent): ");
String dev_per_type = sc.next();

System.out.print("Enter Developer's Number of Projects Managed: ");
int dev_no_of_projects = sc.nextInt();

System.out.println();
```

```
    // Programmer
    System.out.print("Enter Programmer's Name: ");
    String prog_name = sc.next();

    System.out.print("Enter Programmer's Address: ");
    String prog_address = sc.next();

    System.out.print("Enter Programmer's Salary (in $): ");
    double prog_salary = sc.nextDouble();

    System.out.print("Enter Programmer's Experience (in years): ");
    int prog_exp = sc.nextInt();

    System.out.print("Enter Programmer's Performance Type (Bad/Good/Excellent): ");
    String prog_per_type = sc.next();

    System.out.print("Enter Programmer's Number of Projects Managed: ");
    int prog_no_of_projects = sc.nextInt();

    // Calling manager methods
    mgr = new Manager(mgr_name, mgr_address, "Manager", mgr_salary, mgr_exp);
    mgr.info();
    mgr.calculateBonus();
    mgr.generatePerformanceReport(mgr_per_type);
    mgr.manageProjects(mgr_no_of_projects);

    // Calling developer methods
    dev = new Developer(dev_name, dev_address, "Developer", dev_salary, dev_exp);
    dev.info();
    dev.calculateBonus();
    dev.generatePerformanceReport(dev_per_type);
    dev.manageProjects(dev_no_of_projects);

    // Calling programmer methods
    prog = new Programmer(prog_name, prog_address, prog_salary, prog_exp);
    prog.info();
    prog.calculateBonus();
    prog.generatePerformanceReport(prog_per_type);
    prog.manageProjects(prog_no_of_projects);

    sc.close();
  }
}
```

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");
  }
}
```