Common Errors

Chapter 1: Introduction to Computers, Programs, and Java

1. Q: When you compile your program, you receive an error as follows:
   \`
   %javac Welcome.java
   javac not found
   \`
   What is wrong?
   
   A: Two possible reasons: 1. Java is not installed on your system; or 2. Java is installed, but the environment variable PATH is not set. To set the path correctly, see Section 1.3 in Supplement I.B.

2. Q: When you compile your program, you receive an error as follows:
   \`
   %javac Welcome.java
   java: file not found: Welcome.java
   Usage: javac
   use -help for a list of possible options
   \`
   What is wrong?
   
   A: File Welcome.java does not exist in the directory. Check if Welcome.java is in the current directory.

3. Q: When you run your program, you receive an error as follows:
   \`
   %java Welcome
   Exception in thread "main" java.lang.NoClassDefFoundError: Welcome
   \`
   What is wrong?
   
   A: File Welcome.class does not exist. You have to first compile Welcome.java to produce Welcome.class in the current directory.

4. Q: When you run your program, you receive an error as follows:
   \`
   %java Welcome
   Exception in thread "main" java.lang.NoSuchMethodError: main
   \`
   What is wrong?
   
   A: Please check if you have a main method in Welcome.java or you have spelled the method public static void main(String[] args) correctly.

5. Q: When you compile your program, you receive an error as follows:
   \`
   %javac Welcome.java
   Welcome.java:2: cannot find symbol
     symbol  : class String
     location: class Welcome
     public static void main(string[] args) {
     ^
   1 error
   \`
   What is wrong?
   
   A: Java is case-sensitive. string should be spelled as String.

Chapter 2: Elementary Programming

1. Q: What is wrong in the following code?
   \`
   public class Test {
     public static void main(String[] args) {
       i = 1;
       j = i * i * i;
     }
   }
   \`
   A: You will get a compile error that indicates symbol i cannot be resolved. Every variable must be declared before being used.

2. Q: What is wrong in the following code?
   \`
   public class Test {
     public static void main(String[] args) {
       int i;
       j = i * i * i;
     }
   }
   \`
   A: You will get a compile error that indicates variable i is not initialized in line 3.

3. Q: What is wrong in the following code?
   \`
   public class Test {
     public static void main(String[] args) {
       int j = 5;
       System.out.println(j * 4.5);  
     }
   }
   \`
   A: The program compiles and runs fine. However, i is never used. This might be a potential programming error. Most IDE now can alert you of this situation.
1. public class Test {
2    public static void main(String[] args) {
3        int i = 9;
4        int j = 5 / 6 * (i * 4);
5        System.out.println(j);
6    }
7}

A: This type of mistakes is very common for my students. We have covered this repeatedly in the text. Here just to remind you again 5 / 6 is 0, because both 5 and 6 are integers.

2. public class Test {
    public static void main(String[] args) {
        System.out.println(2147483647 + 1);
    }
}

A: Run this program. You will see the output is -2147483648. Why? 2147483647 is the largest int value. Adding 1 to this value causes overflow. It is not a runtime error in Java.

3. public class Test {
    public static void main(String[] args) {
        char ch = 'a';
        System.out.println('a' + 1);
        System.out.println(++ch);
    }
}

A: for 'a' + 1, 'a' is converted into int. so 'a' + 1 is 97 + 1 = 98. For ++ch, the result is 'b'.

Chapter 3: Selections

1. public class Test {
    public static void main(String[] args) {
        int i, j, k;
        i = j = k = 3;
        if (i < j < k)
            System.out.println("i, j, and k are in increasing order");
    }
}

A: i < j evaluates to a boolean value, which cannot be compared with k. The correct expression should be (i < j) && k < j).

2. public class Test {
    public static void main(String[] args) {
        System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1 == 0.0);
    }
}

A: Run the program. You will be surprised to see the output is false. Why? because floating-point numbers are not stored in complete accuracy. You are dealing with approximation for floating-point numbers. Don't compare two floating-point numbers using equality == operators.

3. public class Test {
    public static void main(String[] args) {
        boolean done = false;
        if (done)
            System.out.println("done");
    }
}

A: Run the program. You will see the output is done. Why? you assign true to done. So, the condition is true. To check whether done is true, you should use done == true, or simply

if (done) {
    System.out.println("done");
}

4. public class Test {
    public static void main(String[] args) {
        boolean done = false;
        if (done)
            System.out.println("done");
    }
}

A: Run the program. You will see the output is done. Why? because the if statement ends with

if (done);

The semicolon ends the if statement. The lines

{ System.out.println("done");
}

are not part of the if statement.
5. Q: What is the output of the following code?
   ```java
   public class Test {
      public static void main(String[] args) {
         char ch = 'a';
         switch (ch) {
            case 'a': System.out.print(ch); break;
            case 'b': System.out.print(ch); break;
            case 'c': System.out.print(ch); break;
         }
      }
   }
   ```
   A: Run the program. You will see the output is aaa. Why? There is no break statement for the case statement. Once a case is matched, the statements starting from the matched case are executed until a break statement or the end of the switch statement is reached. This phenomenon is referred to as the fall-through behavior.

Chapter 4: Loops

1. Q: What is wrong in the following code?
   ```java
   public class Test {
      public static void main(String[] args) {
         int i = 0;
         while (i < 10) {
            System.out.println("Welcome to Java!");
            i++;
         }
      }
   }
   ```
   A: i is always 0. This is an infinite loop. This is a common error for while and do-while loops. Make sure the control variable changes to cause the loop to stop eventually.

2. Q: What is wrong in the following code?
   ```java
   public class Test {
      public static void main(String[] args) {
         int i = 0;
         while (i < 10)
            System.out.println("Welcome to Java!");
            i++;
      }
   }
   ```
   A: i++ is not in the body of the while loop. So, this is an infinite loop. If the loop body contains more than one statement, use the braces to group them.

Chapter 5: Methods

1. Q: What is the output of the following code?
   ```java
   public class Test {
      public static void main(String[] args) {
         int x = 1;
         p(x);
         System.out.println("x is " + x);
      }
      public static void p(int x) {
         x = x + 1;
      }
   }
   ```
   A: x is 1. Invoking p(x), the value of x is passed to the parameter in the method. Variable x defined in the main method and x defined in method p are two independent variables.

2. Q: Does Math.sin(90) return the sine value for angle 90 degree?
   A: No. To return the sine value for angle 90 degree, use Math.sin(Math.PI / 2).

Chapter 6: Single-Dimensional Arrays

1. Q: What is wrong in the following code?
   ```java
   public class Test {
      public static void main(String[] args) {
         int[] x = new int[10];
         for (int i = 0; i <= x.length; i++)
            System.out.print(x[i]);
      }
   }
   ```
   A: This is a common error, known as off-by-one. Here the array index is from 0 to 9, but the last i in the loop will be x.length (10). At 10, the index is out of range.

2. Q: What is wrong in the following code?
   ```java
   public class Test {
      public static void main(String[] args) {
         int[] x = new int[10];
         p(x);
      }
      public static void p(int[] list) {
         list[0] = 10;
      }
   }
   ```
   A: The output is 10. When invoking p(x), the reference value of array x is passed to list. The first element in the array is changed in the method.

Chapter 8: Objects and Classes

1. Q: Do local variables have default values? Do instance variables have default values? Do static variables have default values?
   A: Local variables don't have default values. Instance and static variables are data fields and have default values.

2. Q: What is the printout of the following code?
   ```java
   public class Test {
      ```
public void Test() {
    System.out.println("Test");
}

public static void main(String[] args) {
    Test t = new Test();
}

A: None. Note that "public void Test()" is not a constructor. It is a method. The class Test has a default constructor.

3. Q: What is wrong the following code?
   public class Test {
       public static void main(String[] args) {
           String[] x = new String[10];
           System.out.print(x[0].toString());
       }
   }

A: When you create an array of objects. Each element of the array has a null value. So, invoking x[0].toString() causes a NullPointerException.

Chapter 11: Inheritance and Polymorphism

1. Q: What is wrong the following code?
   public class Apple extends Fruit {
   }

   class Fruit {
       public Fruit(String name) {
           System.out.println("Fruit's constructor is invoked");
       }
   }

A: Since no constructor is explicitly defined in Apple, Apple’s default no-arg constructor is defined implicitly. Since Apple is a subclass of Fruit, Apple’s default constructor automatically invokes Fruit’s default constructor. However, String does not have a no-arg constructor, because String has an explicit constructor defined. Therefore, the program cannot be compiled.

2. Q: What is the difference between == and equals()?

A: The == comparison operator is used for comparing two primitive data type values or for determining whether two objects have the same references. The equals method is intended to test whether two objects have the same contents, provided that the method is overridden in the defining class of the objects. The == operator is stronger than the equals method, in that the == operator checks whether the two reference variables refer to the same object.

Chapter 20: Recursion

1. Q: What is wrong the following code?
   public static long factorial(int n) {
       return n * factorial(n - 1);
   }

A: There is no stopping condition. The method runs infinitely.

2. Q: What is wrong the following code?
   public class Test {
       public static void main(String[] args) {
           printLine(2, "Welcome to Java");
       }

       public static void printLine(int n, String message) {
           while (n > 1) {
               System.out.println(message);
               printLine(n - 1, message);
           }
       }
   }

A: while should be replaced by if. With "while", the method runs infinitely. To understand why this happens, draw a stack. First the main method is placed into the stack. Second, println(2, "Welcome to Java") is placed into the stack. Now n is 2. Third, println(1, "Welcome to Java") is placed into the stack. Now n is 1, so the method finished and popped out of the stack. But now you are still in the second method call. In the second method call, n is still 2. The loop-continuation-condition is still true. So, the loop continues. The method will run forever.