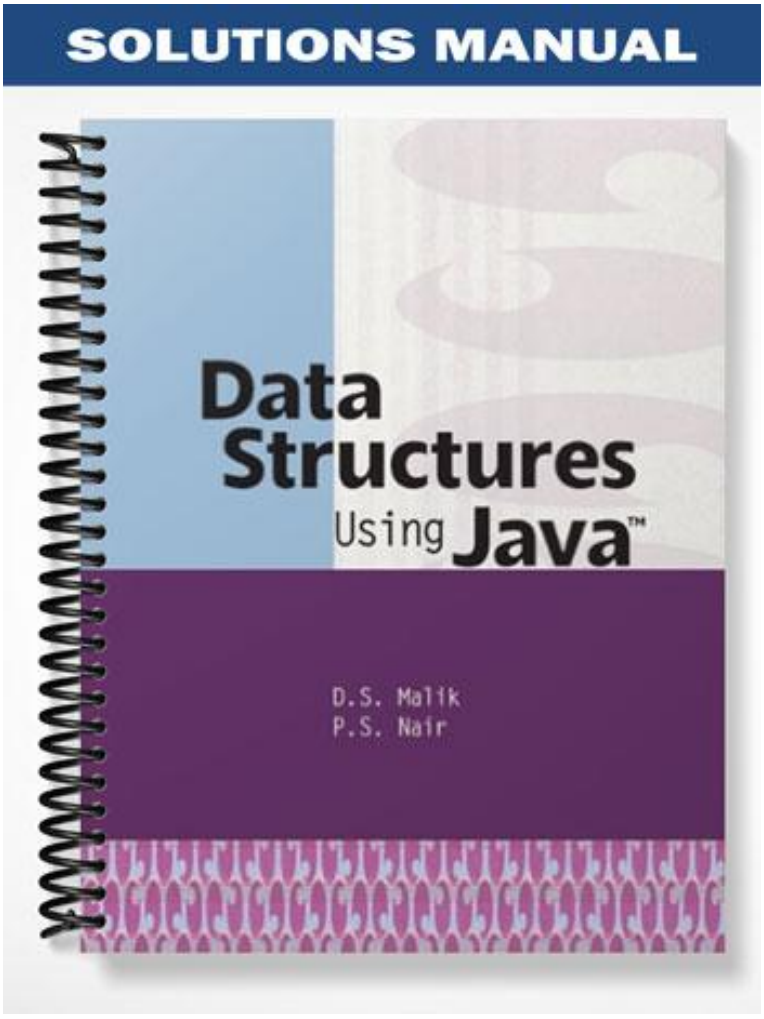
SOLUTIONS MANUAL



Exercise Solutions for Data Structures Using Java

Chapter 1

1. a. true; b. false; c. false; d. false; e. false; f. true; g. false; h. false
2.
   Precondition: The value of `x` must be nonnegative.
   Postcondition: If the value of `x` is nonnegative, the function returns the positive square root of `x`; otherwise, the program terminates.

3. a. $O(n^2)$
   b. $O(n^3)$
   c. $O(n^3)$
   d. $O(n)$

4. 10

5. a. 43
   b. $4n + 3$
   c. $O(n)$

6. -51, -50, -49, -1, 0, 1, 49, 50, 51

7.
   a.
   ```
   int sumSquares(int n)
   {
          int sum = 0;

          for(int j = 1; j <= n; j++)
              sum = sum + j * j;

          return sum;
   }
   ```

   b. This function is of order $O(n)$.

8. The black-box refers to testing the correctness of the program; that is, making sure that the program does what it is supposed to do. In black-box testing, you do not know the internal working of the algorithm or function. You know only what the function does. Black-box testing is based on inputs and outputs.

9. The white-box refers to testing the correctness of the program; that is, making sure that the program does what it is supposed to do. White-box testing relies on the internal structure and implementation of a function or algorithm. The objective is to ensure that every part of the function or algorithm is executed at least once.

10. a. Constructors have no type. Therefore the definition of the constructor with parameters should be:

```
public AA(int a, int b)
{
   x = a;
   y = b;
}
```

b. The type of the method print is missing. The definition of the method print should be:

```
public void print()
{
      System.out.println(one + " " + two);
}
```

11. a. (i) Constructor at Line 1
(ii) Constructor at Line 3
(iii) Constructor at Line 4

b.
```
public CC()
{
  u = 0;
  v = 0;
  w = 0;
}
```

c.
```
public CC(int a)
{
  u = a;
  v = 0;
  w = 0;
}
```

d.
```
public CC(int a, int b)
{
    u = a;
    v = b;
    w = 0;
}
```

e.
```
public CC(int a, int b, double d)
{
    u = a;
    v = b;
    w = d;
}
```

12.
```
Clock mysteryClock = new Clock(7,18,39);
```

13.

06:23:17
06:23:17

14.

a. 6;
b. 2;
c. 2;
d.
```
public void func()
{
     u = 10;
     v = 15.3;
}
```
e.
```
public void print()
```

```
    {
          System.out.println(u + " " + v);
    }
```

f.

```
public xClass()
{
     u = 0;
     v = 0;
}
```

g.

```
public xClass(int a, int b)
{
     u = a;
     v = b;
}
```

h. `x.print();`

i. `XClass t = new XClass(20, 35.0);`

15. In shallow copying, two or more reference variables of the same type refer to the same object.

16. In deep copying each reference variable refers to its own object.

17. Both `aa` and `bb` point to the object `bb`.

18. `06:23:17`

19. The purpose of the copy constructor is to initialize an object, when the object is instantiated, using an existing object of the same type.
20. Java implicitly uses the reference this to refer to both the instance variables and methods of a class.

21. `package dsuj.ch01.strangeClasses;`

22. No.

23 a.

```
public int sum()
{
      return x + y;
}

public void print()
{
      System.out.println("x = " + x + ", y = " + y);
}

public  TestClass()
{
    x = 0;
    y = 0;
}

public TestClass(int a, int b)
{
```

```
        x = a;
        y = b;
    }
```

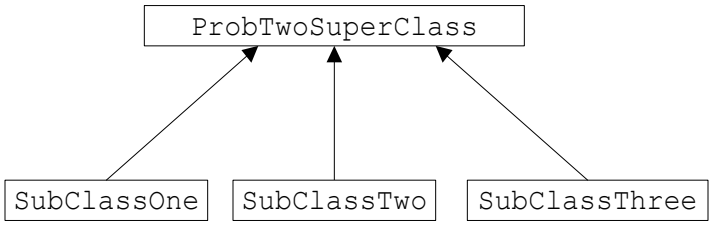
b. (One possible solution. We assume that the `class TestClass` is in the package `dsuj.ch01.Exercises`.)

```
import dsuj.ch01.Exercises.*;

public class Exercise23
{
   public static void main(String[] args)
   {
      TestClass one = new TestClass();
      TestClass two = new TestClass(4,5);

      one.print();
      two.print();
   }
}
```

Chapter 2

1. a. true; b. false; c. true; d. false; e. true; f. false; g. false; h. true; i. false; j. true; k. true; l. false
2. 

3. Some of the data members that can be added to the `class Employee` are: `department`, `salary`, `employeeCategory` (such as supervisor and president), and `employeeID`. Some the member functions are: `setInfo`, `getSalary`, `getEmployeeCategory`, `setSalary`.

4. The private members of a `class` are `private`; they cannot be directly accessed by the member functions of the derived class. The `protected` members of the base class can be directly accessed by the member functions of the derived class

5. In overloading a method, both methods are members of the same class; but have different formal parameter list. In overriding, you are redefining a method in a subclass. Thus two methods appear in two different classes and further, the formal parameter list of both remains the same.

6. In a constructor of a subclass and during the overriding of a method in a subclass.
7. The statement :

```
class BClass AClass
```

should be:

```
class BClass extends AClass
```

Also missing) in `System.out.println` statement. Moreover, variables `u` and `v` are `private` in `class AClass`, and cannot be accessed directly in `class BClass`.

8. (a) False
   (b) i. Valid.
   ii. Invalid: a is a private data member of the class. It cannot be directly accessed outside the class. b is a private data member of the class. It cannot be directly accessed outside the class. Also, yObject and xObject need to be initialized.
   iii. Invalid: `a` and `b` are private in `class YClass` and cannot be accessed directly accessed in `class XClass`.
   iv. Invalid: `a, b,` and `z` are private data members. They cannot be accessed directly by the objects `xObject` and `yObject`. Also, `xObject` and `yObject` need to be initialized.

9.

```
a. public YClass()
   {
      a = 0;
      b = 0;
   }
b. public XClass()
```

```
    {
       super(0,0);
       z = 0;
    }
 c. public void two(int u, int v)
    {
       a = u;
       b = v;
    }
```

10. `aObject` need to be initilized.
11. a.

```
public void setData(int a, int b, int c)
{
      super.setData(a, b);
      z = c;
}
```

b.

```
public void print()
{
      super.print();
      System.out.println(z);
}
```

12.

```
2 This is superclass
SubClass: 7a
10 Hello Super
```

13. A statement such as `rectRef instanceof BoxShape` returns true if the reference variable rectRef points to a BoxShape object.
14. An abstract method is a method that has only the heading with no body. Moreover, the heading of an abstract method contains the reserved word abstract and ends with a semicolon.

15. (a)

```
Entering the try block.
Exception: Lower limit violation.
After the catch block
```

(b)

```
Entering the try block.
Exiting the try block.
After the catch block
```

16. (a)

```
Entering the try block.
Exception: Lower limit violation.
After the catch block
```

(b)

```
Entering the try block.
Exception: / by zero
After the catch block
```

(c)

```
Entering the try block.
Exiting the try block.
After the catch block
```

(d)

```
Entering the try block.
```

```
Exception: / by zero
After the catch block
```

17.

```
catch(Exception e)
{
  if (e instanceof ArithmeticException)
  {
      System.out.println("Exception: " + e.getMessage());
      result = 110;
  }
  else
      System.out.println("Exception: " + e.getMessage());
 }
```

replaces the two catch blocks.

18.

```
public class TornadoException extends Exception
{

      public TornadoException()
      {
          super("Tornado: Take cover immediately");
      }

      public TornadoException(int m)
      {
          super("Tornado: "+ m + "miles away; and approaching!");
      }
}
```

19.

```
import java.io.*;

public class Test
{

      public static void main (String[] args)
      {
            int i = 8;

            try
            {

                  if (i < 5) throw new TornadoException();
                  else throw new TornadoException(i);

            }
            catch(TornadoException e)
            {
                  System.out.println(e.getMessage());
            }
      }
}
```

20. Default constructor:

```
Immediate attention required!
MyException thrown!
```

Non-default constructor

```
Attention required!
Msg
```