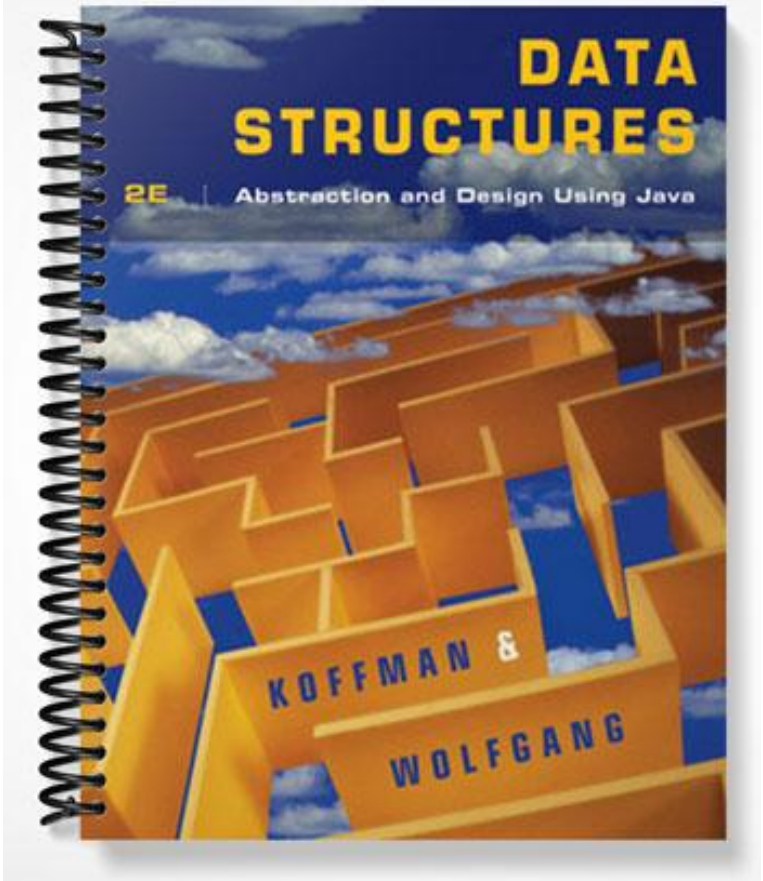
SOLUTIONS MANUAL



Chapter	Section	No
1	1	1

Question

Define an interface named `Resizable` with just one abstract method, `resize`, that is a **`void`** method with no parameter.

Solution

```
//Solution to programming exercise 1, Section 1, Chapter 1
//File: \KW\CH01\Resizable.java startLine: 0 endLine 14
package KW.CH01;

/**
 *  Interface to define the method resize
 */
public interface Resizable {

    /**
     *  Method to resize the object
     */
    void resize();
}
```

Chapter	Section	No
1	1	2

Question

Write a Javadoc comment for the following method of a class `Person`. Assume that class `Person` has two `String` data fields `lastName` and `firstName` with the obvious meanings. Provide preconditions and postconditions if needed.

```
Public int compareTo(Person per) {
    if (lastName.equals(per.lastName))
        return firstName.compareTo(per.firstName);
    else
        return lastName.compareTo(per.lastName);
}
```

Solution

```
//Solution to programming exercise 2, Section 1, Chapter 1
//File: \KW\CH01\Person.java startLine: 179 endLine 191
    /**
     * Method to compare two Person objects based on name.
     *
     * @param per The Person object to compare this Person object to
     *
     * @return <0 If this Person is less than (alphabetically before)
     *         per 0 If this Person is equal to per >0 If this
     *         Person is greater than (alphabetically after) per
     */
```

Chapter	Section	No
1	1	3

Question

Write a Javadoc comment for the following method of class `Person`. Provide preconditions and postconditions if needed.

```
Public void changeLastName(boolean justMarried, String newLast) {
    if (justMarried)
        lastName = newLast;
}
```

Solution

```
//Solution to programming exercise 3, Section 1, Chapter 1
//File: \KW\CH01\Person.java startLine: 199 endLine 213
    /**
     * Method to set the last name of this Person to a new
     * value provided that justMarried is true.
     *
     * @param justMarried true if this Person's name is to be changed
     *        param newLast The new last name if justMarried is true
     * @param newLast DOCUMENT ME!
     *
     * @post lastName is equal to newLast if justMarried is true
     *       otherwise no change is made to this Person
     */
```

Chapter	Section	No
1	1	4

Question

Write method `verifyPIN` for class `ATMBankAmerica` assuming this class has a data field `pin` (type `String`).

Solution

```
//Solution to programming exercise 4, Section 1, Chapter 1
```

```
//File: \KW\CH01\ATMbankAmerica.java startLine: 6 endLine 22

    /** The user's PIN */
    private String pin;

    /**
     * Verifies a user's PIN.
     * @param pin The user's PIN
     * @return true if the user's PIN is verified
     */
    @Override
    public boolean verifyPIN(String pin) {
        return this.pin.equals(pin);
    }
```

Chapter	Section	No
1	2	1

Question

Write accessor and modifier methods for class `Computer`.

Solution

```
//Solution to programming exercise 1, Section 2, Chapter 1
//File: \KW\CH01\Computer.java startLine: 60 endLine 90
    public String getManufacturer() {
        return manufacturer;
    }

    public String getProcessor() {
        return processor;
    }

    public void setManufacurer(String man) {
        manufacturer = man;
    }

    public void setProcessor(String processor) {
        this.processor = processor;
    }

    public void setRamSize(int ram) {
        ramSize = ram;
    }

    public void setDiskSize(int disk) {
        diskSize = disk;
    }

    public void setProcessorSpeed(double procSpeed) {
        processorSpeed = procSpeed;
    }
```

Chapter	Section	No
1	2	2

Question

Write accessor and modifier methods for class `Notebook`.

Solution

```
//Solution to programming exercise 2, Section 2, Chapter 1
//File: \KW\CH01\Notebook.java startLine: 40 endLine 58
    // Accessor and modifier methods
    public double getScreenSize() {
        return screenSize;
    }

    public double getWeight() {
        return weight;
    }

    public void setScreenSize(double screen) {
        screenSize = screen;
    }

    public void setWeight(double wei) {
        weight = wei;
    }
```

Chapter	Section	No
1	3	1

Question

Write constructors for both classes that allow you to specify only the processor, RAM size, and disk size.

Solution

```
//Solution to programming exercise 1, Section 3, Chapter 1
//File: \KW\CH01\Notebook.java startLine: 34 endLine 39
    public Notebook(String processor, double ram, int disk) {
        this("Default", processor, ram, disk, 2.5, 17, 5.5);
    }
//Solution to programming exercise 1, Section 3, Chapter 1
//File: \KW\CH01\Computer.java startLine: 37 endLine 43
    public Computer(String processor, double ram, int disk) {
        this("Default", processor, ram, disk, 2.5);
    }
```

Chapter	Section	No
1	3	2

Question

Complete the accessor and modifier methods for class `Computer`.

Solution

```
//Solution to programming exercise 2, Section 3, Chapter 1
//File: \KW\CH01\Computer.java startLine: 91 endLine 94
    // See the solution for 1.2.1
```

Chapter	Section	No
1	3	3

Question

Complete the accessor and modifier methods for class `Notebook`.

Solution

```
//Solution to programming exercise 3, Section 3, Chapter 1
//File: \KW\CH01\Notebook.java startLine: 59 endLine 62
    // See the solution for 1.2.2
```

Chapter	Section	No
1	4	1

Question

Write class `Vegetable`. Assume that a vegetable has three double constants: `VEG_PROTEIN_CAL`, `VEG_FAT_CAL`, and `VEG_CARBO_CAL`. Compute the fat percentage as `VEG_FAT_CAL` divided by the sum of all the constants.

Solution

```
//Solution to programming exercise 1, Section 4, Chapter 1
//File: \KW\CH01\Vegetable.java startLine: 0 endLine 51
package KW.CH01;

/**
 * Class to represent a Vegetable
 */
public class Vegetable extends Food {

    /** Calories from protein */
    private static final double VEG_PROTEIN_CAL = 0.35;
    /** Calories from fat */
    private static final double VEG_FAT_CAL = 0.15;
    /** Calories from carbohydrates */
    private static final double VEG_CARBO_CAL = 0.50;
    /** The name of the vegetable */
    private String name;

    /**
     * Constructor
     * @param name The name of the vegetable
     */
    public Vegetable(String name) {
        this.name = name;
        setCalories(VEG_PROTEIN_CAL + VEG_FAT_CAL + VEG_CARBO_CAL);
    }

    /**
     * Calculates the percent of protein in a Food object.
     * @return The percentage of protein
     */
    public double percentProtein() {
        return VEG_PROTEIN_CAL / getCalories();
    }

    /**
     * Calculates the percent of fat in a Food object.
     * @return The precentage of fat
```

```
     */
    public double percentFat() {
        return VEG_FAT_CAL / getCalories();
    }

    /**
     * Calculates the percent of carbohydrates in a Food object.
     * @return The percentage of carbohydrates
     */
    public double percentCarbohydrates() {
        return VEG_CARBO_CAL / getCalories();
    }
}
```

Chapter Section No
1 4 2

Question

Earlier we discussed a `Computer` class with a `Notebook` class as its only subclass. However, there are many different kinds of computers. An organization may have servers, mainframes, desktop PCs, and notebooks. There are also personal data assistants and game computers. So it may be more appropriate to declare class Computer as an abstract class that has an actual subclass for each category of computer. Write an abstract class Computer that defines all the methods shown earlier and declares an abstract method with the signature `costBenefit(double)` that returns the cost-benefit (type `double`) for each category of computer.

Solution

```
//Solution to programming exercise 2, Section 4, Chapter 1
//File: \KW\CH01\AbstractComputer.java startLine: 0 endLine 99
package KW.CH01;

/**
 * Class that represents a computer.
 */
public abstract class AbstractComputer {
    // Data Fields

    private String manufacturer;
    private String processor;
    private double ramSize;
    private int diskSize;
    private double processorSpeed;

    // Methods
    /**
     * Initializes a Computer object with all properties specified.
     * @param man The computer manufacturer
     * @param processor The processor type
     * @param ram The RAM size
     * @param disk The disk size
     * @param procSpeed The processor speed
     */
    public AbstractComputer(String man, String processor, double ram,
            int disk, double procSpeed) {
        manufacturer = man;
        this.processor = processor;
        ramSize = ram;
        diskSize = disk;
        processorSpeed = procSpeed;
    }

    public double computePower() {
        return ramSize * processorSpeed;
    }

    public double getRamSize() {
        return ramSize;
    }

    public double getProcessorSpeed() {
        return processorSpeed;
    }

    public int getDiskSize() {
        return diskSize;
    }

    // Insert other accessor and modifier methods here.
    public String getManufacturer() {
        return manufacturer;
    }

    public String getProcessor() {
        return processor;
```

```
    }

    public void setManufacurer(String man) {
        manufacturer = man;
    }

    public void setProcessor(String processor) {
        this.processor = processor;
    }

    public void setRamSize(int ram) {
        ramSize = ram;
    }

    public void setDiskSize(int disk) {
        diskSize = disk;
    }

    public void setProcessorSpeed(double procSpeed) {
        processorSpeed = procSpeed;
    }

    public String getRamSize(String s) {
        return Double.toString(ramSize);
    }

    @Override
    public String toString() {
        String result = "Manufacturer: " + manufacturer + "\nCPU: "
                + processor + "\nRAM: " + ramSize + " gigabytes"
                + "\nDisk: " + diskSize + " gigabytes"
                + "\nProcessor speed: " + processorSpeed + " gigahertz";
    }

    /**
     * Compute the cost/benefit of this computer.
     * @return the cost/benefit of this computer.
     */
    public abstract double costBenefit();
}
```

Chapter	Section	No
1	5	1

Question

Write an equals method for class `Computer` (Listing 1.2).

Solution

```
//Solution to programming exercise 1, Section 5, Chapter 1
//File: \KW\CH01\Computer.java startLine: 130 endLine 164
    /**
     * Determine if this Computer is equal to the other
     * object
     *
     * @param obj The object to compare this Computer to
     *
     * @return true If the other object is of type Computer and all
     *         data fields are equal
     */
    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }

        if (obj == null) {
            return false;
        }

        if (obj.getClass() == this.getClass()) {
            Computer other = (Computer) obj;

            return getManufacturer().equals(other.getManufacturer())
                    && getProcessor().equals(other.getProcessor())
                    && (getRamSize() == other.getRamSize())
                    && (getDiskSize() == other.getDiskSize())
                    && (getProcessorSpeed() == other.getProcessorSpeed());
        } else {
            return false;
        }
    }
```

Chapter	Section	No
1	5	2

Question

Write an equals method for class `Notebook` (Listing 1.4).

Solution

```
//Solution to programming exercise 2, Section 5, Chapter 1
//File: \KW\CH01\Notebook.java startLine: 63 endLine 99
    /**
     * Determine if this Notebook is equal to the other
     * object
     *
     * @param obj The object to compare this Notebook to
     *
     * @return true If the other object is of type Notebook and all
     *         data fields are equal
     */
    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }

        if (obj == null) {
            return false;
        }

        if (obj.getClass() == this.getClass()) {
            Notebook other = (Notebook) obj;

            return getManufacturer().equals(other.getManufacturer())
                    && getProcessor().equals(other.getProcessor())
                    && (getRamSize() == other.getRamSize())
                    && (getDiskSize() == other.getDiskSize())
                    && (getProcessorSpeed() == other.getProcessorSpeed())
                    && (getScreenSize() == other.getScreenSize())
                    && (getWeight() == other.getWeight());
        } else {
            return false;
        }
    }
```

Chapter	Section	No
1	5	3

Question

Write an equals method for the following class. What other equals methods should be defined?

```
public class Airplane {
// Data Fields
Engine eng;
Rudder rud;
Wing[] wings = new Wing[2];
...
}
```

Solution

```
//Solution to programming exercise 3, Section 5, Chapter 1
//File: \KW\CH01\Airplane.java startLine: 11 endLine 34
    @Override
    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }

        if (obj == null) {
            return false;
        }

        if (obj.getClass() == this.getClass()) {
            Airplane other = (Airplane) obj;

            return eng.equals(other.eng) && rud.equals(other.rud)
                    && wings[0].equals(other.wings[0])
                    && wings[1].equals(other.wings[1]);
        } else {
            return false;
        }
    }
    // equals methods need to be defined for classes Engine, Rudder, and Wing
```

Chapter	Section	No
1	8	1

Question

Write class `Circle`.

Solution

```
//Solution to programming exercise 1, Section 8, Chapter 1
//File: \KW\CH01\Circle.java startLine: 0 endLine 76
package KW.CH01;

import java.util.Scanner;

/**
 * Represents a circle.
 * Extends Shape.
 */
public class Circle extends Shape {
    // Data Fields

    /** The radius of the circle */
    private double radius = 0;

    // Constructors
    /** Constructs a default Circle */
    public Circle() {
        super("Circle");
    }

    /**
     * Constructs a circle of the specified size.
     * @param radius the radius
     * @param height the height
     */
    public Circle(double radius) {
        super("Circle");
        this.radius = radius;
    }

    // Methods
    /**
     * Get the radius.
     * @return The width
     */
    public double getRadius() {
        return radius;
    }

    /**
     * Compute the area.
     * @return The area of the circle
     */
    @Override
    public double computeArea() {
        return Math.PI * radius * radius;
    }

    /**
     * Compute the perimeter.
     * @return The perimeter of the circle
     */
    @Override
    public double computePerimeter() {
        return 2 * Math.PI * radius;
    }

    /** Read the attributes of the circle. */
    @Override
    public void readShapeData() {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the radius of the Circle");
        radius = in.nextDouble();
    }

    /**
     * Create a string representation of the circle.
     * @return A string representation of the circle
     */
    @Override
    public String toString() {
        return super.toString() + ": radius is " + radius;
    }
}
```

Chapter	Section	No
1	8	2

Question

Write class `RtTriangle`.

Solution

```
//Solution to programming exercise 2, Section 8, Chapter 1
//File: \KW\CH01\RtTriangle.java startLine: 0 endLine 98
package KW.CH01;

import java.util.Scanner;

/**
 * Represents a Right Triangle. Extends Shape.
 */
public class RtTriangle extends Shape {
    // Data Fields

    /** The base of the RtTriangle. */
    private double base = 0;
    /** The height of the RtTriangle. */
    private double height = 0;

    // Constructors
    /** Constructs a default RtTriangle */
    public RtTriangle() {
        super("Right Triangle");
    }

    /**
     * Constructs a Right Triangle of the specified size.
     * @param base the base
     * @param height the height
     */
    public RtTriangle(double base, double height) {
        super("Right Triangle");
        this.base = base;
        this.height = height;
    }

    // Methods
    /**
     * Get the base.
     *
     * @return The base
     */
    public double getBase() {
        return base;
    }

    /**
     * Get the height.
     *
     * @return The height
     */
    public double getHeight() {
        return height;
    }

    /**
     * Compute the area.
     *
     * @return The area of the RtTriangle
     */
    @Override
    public double computeArea() {
        return 0.5 * height * base;
    }

    /**
     * Compute the perimeter.
     *
     * @return The perimeter of the RtTriangle
     */
    @Override
    public double computePerimeter() {
        double hyp = Math.sqrt((base * base) + (height * height));

        return hyp + base + height;
    }

    /**
```

```
     * Read the attributes of the Right Triangle.
     */
    @Override
    public void readShapeData() {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the base of the Right Triangle");
        base = in.nextDouble();
        System.out.println("Enter the height of the Right Triangle");
        height = in.nextDouble();
    }

    /**
     * Create a string representation of the RtTriangle.
     *
     * @return A string representation of the RtTriangle
     */
    @Override
    public String toString() {
        return super.toString() + ": base is " + base
                + ", height is " + height;
    }
}
```

Chapter Section No
2 1 1

Question

Write the following static method:

```
/** Replaces each occurrence of oldItem in aList with newItem. */
public static void replace(ArrayList<String> aList, String oldItem,
    String newItem)
```

Solution

```
//Solution to programming exercise 1, Section 1, Chapter 2
//File: \KW\CH02\Exercises.java startLine: 9 endLine 28
  /**
   * Method to replace each occurence of oldItem with newItem
   * in an ArrayList<String>
   * @param aList The arraylist in which items are to be replaced
   * @param oldItem The item to be replaced
   * @param newItem The item to rplace oldItem
   * @post All occurences of oldItem have been replaced with newItem
   */
  public static void replace(ArrayList<String> aList,
      String oldItem,
      String newItem) {
    int index = aList.indexOf(oldItem);
    while (index != -1) {
      aList.set(index, newItem);
      index = aList.indexOf(oldItem);
    }
  }
```

Chapter Section No
2 1 2

Question

Write the following static method:

```
/** Deletes the first occurrence of target in aList. */
public static void delete(ArrayList<String> aList, String target)
```

Solution

```
//Solution to programming exercise 2, Section 1, Chapter 2
//File: \KW\CH02\Exercises.java startLine: 29 endLine 48
  /**
   * Method to delete the first occurence of target from an
   * ArrayList<String>
   * @param aList Thie array list to remove target from
   * @param target The object to be removed
   * @post First occurence of target is no longer in aList
   */
```

```
public static void delete(ArrayList<String> aList, String target) {
   int index = aList.indexOf(target);
   if (index != -1) {
      aList.remove(index);
   }
}
// Note this could also be written as
// public static delete(ArrayList<String> aList, String target) {
//    aList.remove(target);
// }
```

Chapter	Section	No
2	2	1

Question

Write a method `addOrChangeEntry` for a class that has a data field `theDirectory` (type `ArrayList<DirectoryEntry>`) where class `DirectoryEntry` is described just before the exercises. Assume class `DirectoryEntry` has an accessor method `getNumber` and a setter method `setNumber`.

```
/** Add an entry to theDirectory or change an existing entry.
    @param aName The name of the person being added or changed
    @param newNumber The new number to be assigned
    @return The old number, or if a new entry, null
*/
public String addOrChangeEntry(String aName, String newNumber) {
```

Solution

```
//Solution to programming exercise 1, Section 2, Chapter 2
//File: \KW\CH02\PhoneDirectory.java startLine: 17 endLine 41
    /**
     * Add an entry or change an existing entry.
     * @param name The name of the person being added or changed.
     * @param newNumber The new number to be assigned.
     * @return The old number, or if a new entry, null.
     */
    public String addOrChangeEntry(String name,
            String newNumber) {
        int index = 0;
        String oldNumber = null;
        while (index < theDirectory.size()
                && !theDirectory.get(index).getName().equals(name)) {
            index++;
        }
        if (index < theDirectory.size()) {
            oldNumber = theDirectory.get(index).getNumber();
            theDirectory.get(index).setNumber(newNumber);
        } else {
            theDirectory.add(new DirectoryEntry(name, newNumber));
        }
        return oldNumber;
    }
```

Chapter	Section	No
2	2	2

Question

Write a `removeEntry` method for the class in programming exercise 1. Use `ArrayList` methods `indexOf` and `remove`.

```
/** Remove an entry.
    @param aName The name of the person being removed
    @return The entry removed, or null if there is no entry for aName
*/
public Entry removeEntry(String aName) {
```

Solution

```
//Solution to programming exercise 2, Section 2, Chapter 2
//File: \KW\CH02\PhoneDirectory.java startLine: 43 endLine 64
    /**
     * Remove an entry
     * @param aName The name of the person to be removed
     * @return The entry removed, or null if there is no entry for aName
     * @post The directory no longer contains an entry for aName
     */
    public DirectoryEntry removeEntry(String aName) {
        int index = 0;
        String oldNumber = null;
        while (index < theDirectory.size()
                && !theDirectory.get(index).getName().equals(aName)) {
            index++;
        }
        if (index < theDirectory.size()) {
            return theDirectory.remove(index);
        } else {
            return null;
        }
    }
```

Chapter	Section	No
2	3	1

Question

Implement the `indexOf` method of the `KWArrayList<E>` class.