Solutions to Quick Check Questions

7

Defining Your Own Classes—Part 2

7.1 Returning an Object from a Method

1. What's wrong with the following declaation?

```
class Question {
   Person student;

   public void getStudent(  ) {

      return student;
   }

   ...
}
```

The return type for the getStudent method must be Person as in

```
public Person getStudent( ) {

   return student;
}
```

2. Define a Vehicle class. It has a data member owner of type Person. Include an accessor to retrieve the owner person and a mutator to set the owner.

Answer:

```
class Vehicle {

    private Person owner;

    public Person getOwner( ) {
        return owner;
    }

    public void setOwner(Person person) {
        owner = person;
    }
}
```

7.2 The Reserved Word this

1. Write a single statement to express the following operations on fractions using the methods from the Fraction class:

```
f5 = (f1 + f2) / (f3 - f4)
```

Answer:

```
//assume f1, ..., f5 are declared and
//assigned values

f5 = f1.add(f2).divide(f3.minus(f4));
```

2. If the add method is defined thus

```
public void add(Fraction frac) {

    int a, b, c, d;

    a = this.getNumerator();   //get this fraction's
    b = this.getDenominator(); //num and denom

    c = frac.getNumerator();   //get frac's num
    d = frac.getDenominator(); //and denom

    setNumerator(a*b + c*b);   //updates this
    setDenominator(b*d);    //fraction's num and denom
```

```
}
```

why is it wrong to use the method as

```
f3 = f1.add(f2);
```

Because this add *method is a void method. The method does not return a Fraction object. Rather, it adds the argument Fraction object to the receiving Fraction object. The correct would be*

```
f1.add(f2);
```

3. Write statements to assign the sum of fractions f1 and f2 to fraction f3 using the add method defined in Quick Check question 2 above.

```
f1.add(f2);
f3 = f1;
```

The Fraction object f3 becomes the sum of f1 and f2, but the value of f1 will change also. If we do not want f1 to change (which probably is the case for most situations), then we write

```
f3 = new Fraction(f1);
f3.add(f2);
```

7.3 Overloaded Methods and Constructors

1. Are there any conflicts in the following three constructors for ClassX to be valid?

```
public ClassX( int X ) {

    ...
}

public ClassX( float X ) {

    ...
}
```

```
public ClassX( int Y ) {

    ...
}
```

Yes, the first and the third constructors have the same signature so they cannot be overloaded.

2. Define a Student class. A Student has a name. Define two constructors, one with no argument and another with the name as its argument. Initialize the name to a default value Unknown for the zero-argument constructor.

   Answer:

```
class Student {

    private String name;

    public Student( ) {

        name = "Unknown"; // or this("Unknown");
    }

    public Student(String name) {
        this.name = name;
    }
    ...
}
```

3. Rewrite the following constructors, so the first one calls the second one:

```
public ClassOne(int alpha) {
    this.alpha = alpha;
    this.beta  = 0;
}

public ClassOne(int alpha, int beta) {

    this.alpha = alpha;
    this.beta  = beta;
}
```

Answer:

```
public ClassOne(int alpha) {
   this(alpha, 0);
}
```

7.4 Class Variables and Methods

No Quick Check Questions.

7.5 Call-by-Value Parameter Passing

1. What is the name of the scheme used in Java to pass arguments to a method?

   Call-by-value (also called as Pass-by-value).

2. What is the output from the following code?

```
class Question {
   private int one;

   public void myMethod( int one ) {
      this.one = one;
      one = 12;
   }
}

class Test {
   public static void main(String[] arg) {
      int one = 30;

      Question q = new Question();
      q.myMethod(one);

      System.out.println(one);
   }
}
```

Answer:

```
30
```

7.6 Organizing Classes into a Package

No Quick Check Questions.

7.7 Using Javadoc Comments for Class Documentation

1. Add javadoc comments to the following class:

```
class Instructor {
   private String name;

   public void setName(Sring name) {
      this.name = name;
   }

   public String getName( ) {
      return name;
   }
}
```

Answer:

```
/**
 * The class to model an instructor
 *
 * @author Dr. Caffeine
 */
class Instructor {

      /** the name of this instuctor */
      private String name;

      /**
       * Assigns the name to this instructor
       *
       * @param name the name to assign
       */
      public void setName(Sring name) {
         this.name = name;
      }

      /**
       * Returns this instructor's name
       *
       * @return the name of this instructor
       */
      public String getName( ) {
```

```
            return name;
        }
    }
```

2. What is the purpose of @author tag?

   The @author tag is used to record the author of the class.

7.8 The Complete Fraction Class

No Quick Check Questions

7.9 Sample Development:Library Overdue Checker

No Quick Check Questions