

Solutions to Quick Check Questions

4

Defining Your Own Classes

4.1 Defining Instantiable Classes

1. The following is invalid. Why?

```
public void myMethod( int one )
{
    return (one + one) ;
}
```

A void method cannot return a value, so a return statement inside a void method does not make sense and invalid.

2. Which of the following methods are class methods?

```
1 —▶ public static void one( ) { ... }
2 —▶ private int two( ) { ... }
3 —▶ private static int three( ) { ... }
4 —▶ public float four( ) { ... }
```

1. *Class*

2. *Instance*

3. *Class*

4. Instance

3. Define a Student class. A Student has name. There are two methods, one called identify that returns the student's name and another called assign that assigns the student's name.

```
class Student {  
  
    private String name;  
  
    public String identify( ) {  
        return name;  
    }  
  
    public void assign(String sname) {  
        name = sname;  
    }  
}
```

4.2 Instantiable Classes and Constructors



Quick Check

1. Which of the following constructors are invalid?

```
1 —> public int ClassA( int one )  
    {  
        ...  
    }  
  
2 —> public ClassB( int one, int two )  
    {  
        ...  
    }  
  
3 —> void ClassC( )  
    {  
        ...  
    }
```

1. *Invalid. The constructor does not have a return type.*

2. *Valid.*

3. *Invalid. The void modifier is not valid with the constructor.*

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

```
1 —▶ public ClassX( int X )
      {
          ...
      }
```

```
2 —▶ public ClassX( float X )
      {
          ...
      }
```

```
3 —▶ public ClassX( int Y )
      {
          ...
      }
```

Constructor 1 and 3 will conflict because both declare one parameter of type int.

3. Define a Student class. A Student has name. There are two methods, one called `identify` that returns the student's name and another called `assign` that assigns the student's name. Define two constructors, one with no argument and another with the name as its argument.

```
class Student {
    private static final String NO_NAME =
        "No Name Assigned";

    private String name;

    public void Student( ) {
        this(NO_NAME);
    }

    public void Student(String sname) {
        assign(sname);
    }

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

```

        public void assign(String sname) {
            name = sname;
        }
    }

```

4.3 Information Hiding and Visibility Modifiers

1. If the data member `feeRate` is private, is the following statement valid?

```

CurrencyConverter converter
    = new CurrencyConverter();
double amount = converter.feeRate;

```

No. Direct access is not valid for a private data member.

2. Suppose you wrote down important information such as your bank account number, student registration ID, and so forth, on a single sheet of paper. Will this sheet be declared private, and kept in your desk drawer, or public, and placed next to the dorm's public telephone?

Should be declared private.

4.4 Local Variables, Return Values, and Parameter Passing

1. Identify the local variables, parameters, and data members in the following method `myMethod`:

```

class Question1 {
    private int one;
    private int four;

    public void myMethod( int one ) {
        double two = 2;
        int three = 3;
        return one + two * four;
    }
}

```

local variables: two, three

parameters: one

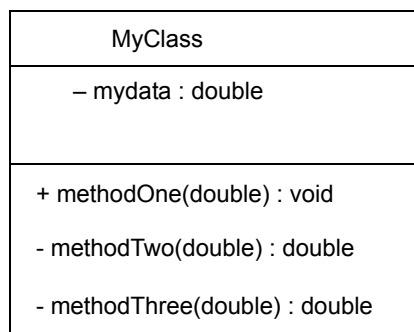
data membes: four

2. What is the problem with the following method?

```
public void problem( ) {
    int one = 1;
    return one + 45;
}
```

The return type is declared as void, but it should be int.

3. Identify the private methods from the following diagram:



methodTwo and methodThree are private methods.

4. Complete the following constructor :

```
class Test {
    private double score;

    public Test(double score) {
        //assign the value of parameter to
        //the data member
    }
}
```

public Test(double score) {

this.score = score;

```
}

```

4.5 Accessors, Mutators, and Overloaded Methods

1. Rewrite the following class with using the optional `this` wherever it is allowed:.

```
class One {
    private int var1;
    private int var2;

    public m1( ) {
        var1 = 20;
        m2(var1);
    }

    public m2(int x) {
        var2 = x * 2;
    }
}
```

Answer:

```
class One {
    private int var1;
    private int var2;

    public m1( ) {
        this.var1 = 20;
        this.m2(this.var1);
    }

    public m2(int x) {
        this.var2 = x * 2;
    }
}
```

2. Why is it not considered a good programming practice to write the `setPound` method as

```
public void setPound(int lb) {

    int oz = (int) (gram/LB_TO_GRAM
        - Math.floor(gram/LB_TO_GRAM));

    weight = lb * LB_TO_GRAM + oz * OZ_TO_GRAM;
}
```

Because it duplicates the same code of the getOunce and convertToGram methods. Avoid duplication of code whenever possible and appropriate.

4.6 Passing and Returning Objects

No Quick Check Questions.

4.7 Modularizing the Input Routine Functionality

1. Using the InputHandler class, write a code fragment to input the user's age.

```
InputHandler input = new InputHandler();  
  
int age = input.getInteger("Enter age:");
```

2. What is wrong with the following code?

```
InputHandler in = new InputHandler();  
int roomNumber = in.getString("Your room #:");
```

The assignment statement has type mismatch. Variable roomNumber is declared int, but the getString method, which returns a String, is called.

4.8 (Optional) Organizing Classes into a Package

No Quick Check Questions.

4.9 Sample Development: Defining and Using Instantiable Classes

No Quick Check Questions.

