Chapter 7
Object-Oriented Design

Java Software Solutions

Foundations of Program Design

9th Edition, Global Edition

John Lewis
William Loftus

Outline

Software Development Activities

Static Variables and Methods

Class Relationships

⇨ **Interfaces**

Enumerated Types Revisited

Method Design and Overloading

Testing

GUI Design

Mouse and Keyboard Events

Interfaces

- An ***abstract method*** is a method header without a method body. For example,

```
public abstract String toString();
```

- A Java ***interface*** is a collection of abstract methods and constants
- An abstract method can be declared using the modifier `abstract`, but because all methods in an interface are abstract, usually it is left off
- An interface is used to establish a set of methods that a class will implement

Interfaces

None of the methods in an interface are given a definition (body)

`interface` is a reserved word

```
public interface Doable
{
   public void doThis();
   public int doThat();
   public void doThis2(double value, char ch);
   public boolean doTheOther(int num);
}
```

A semicolon immediately follows each method header

Interfaces

- An interface cannot be instantiated
- Methods in an interface have public visibility by default
- A class formally implements an interface by:
  - stating so in the class header
  - providing implementations for every abstract method in the interface
- If a class declares that it implements an interface, it must define all methods in the interface

Interfaces

`implements` is a reserved word

```
public class CanDo implements Doable
{
   public void doThis()
   {
      // whatever
   }

   public void doThat()
   {
      // whatever
   }

   // etc.
}
```

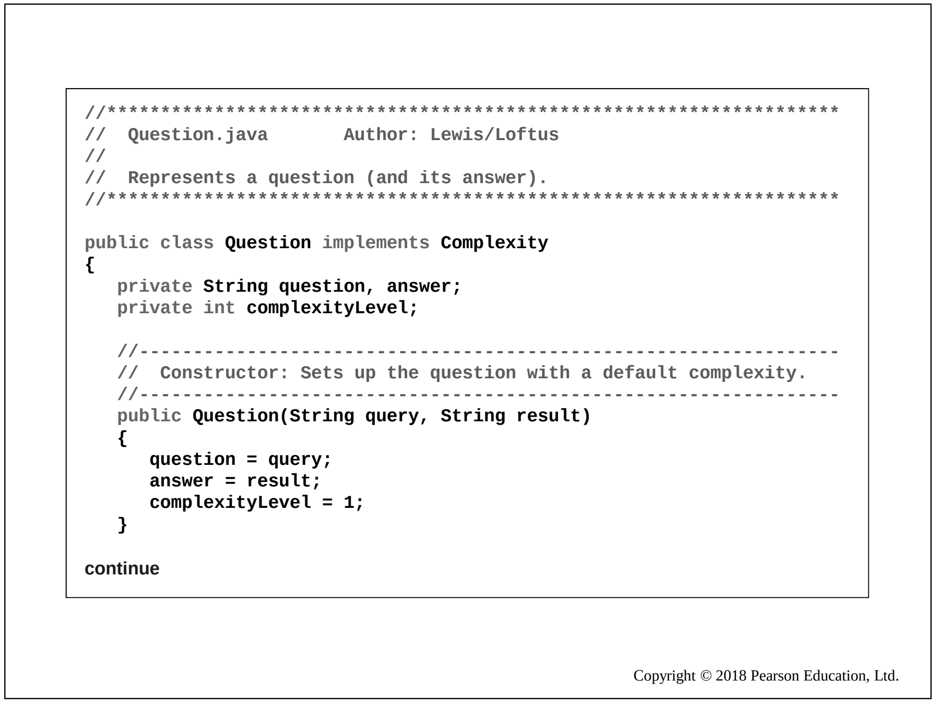
Each method listed in `Doable` is given a definition

Interfaces

- In addition to (or instead of) abstract methods, an interface can contain constants
- When a class implements an interface, it gains access to all its constants
- A class that implements an interface can implement other methods as well
- See `Complexity.java`
- See `Question.java`
- See `MiniQuiz.java`

```
//*********************************************************************
//  Complexity.java       Author: Lewis/Loftus
//
//  Represents the interface for an object that can be assigned an
//  explicit complexity.
//*********************************************************************

public interface Complexity
{
   public void setComplexity(int complexity);
   public int getComplexity();
}
```

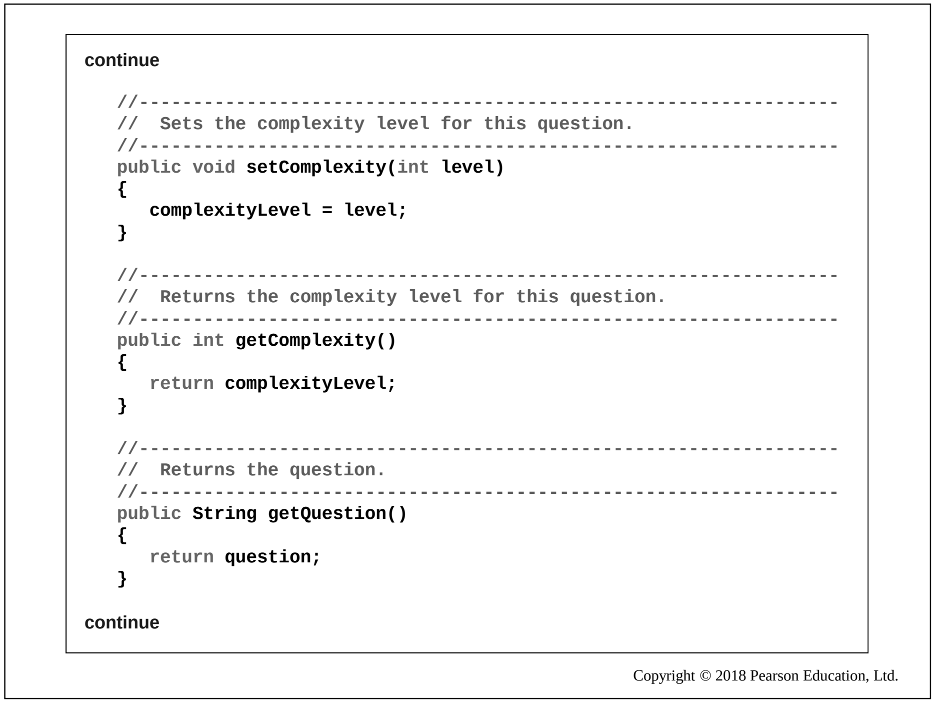


```
//********************************************************************
//  Question.java       Author: Lewis/Loftus
//
//  Represents a question (and its answer).
//********************************************************************

public class Question implements Complexity
{
   private String question, answer;
   private int complexityLevel;

   //-----------------------------------------------------------------
   //  Constructor: Sets up the question with a default complexity.
   //-----------------------------------------------------------------
   public Question(String query, String result)
   {
      question = query;
      answer = result;
      complexityLevel = 1;
   }
```

continue



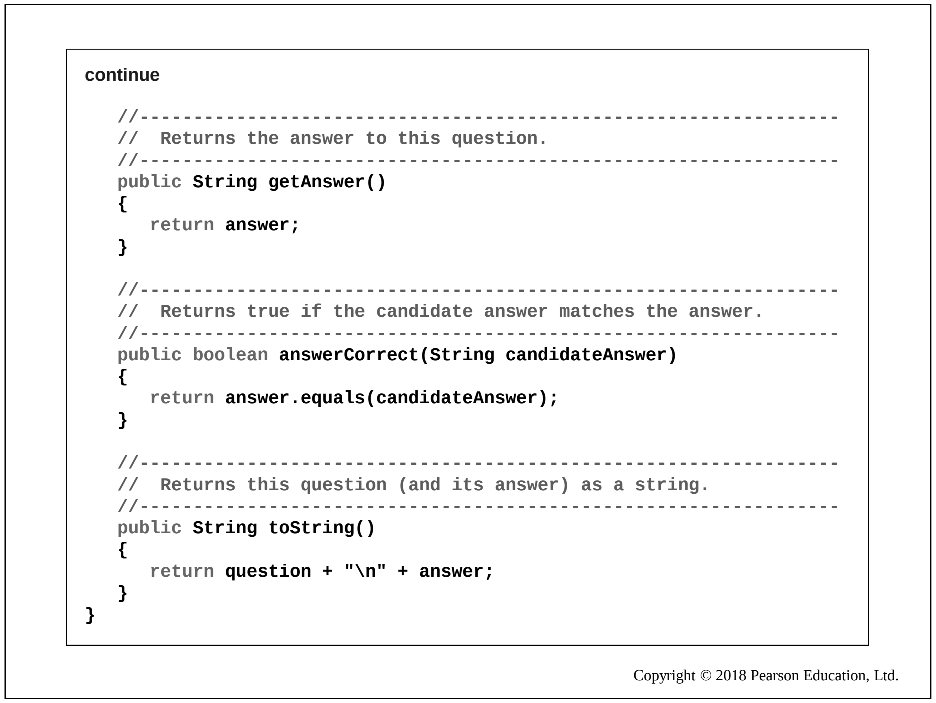
continue

```
   //-----------------------------------------------------------------
   //  Sets the complexity level for this question.
   //-----------------------------------------------------------------
   public void setComplexity(int level)
   {
      complexityLevel = level;
   }

   //-----------------------------------------------------------------
   //  Returns the complexity level for this question.
   //-----------------------------------------------------------------
   public int getComplexity()
   {
      return complexityLevel;
   }

   //-----------------------------------------------------------------
   //  Returns the question.
   //-----------------------------------------------------------------
   public String getQuestion()
   {
      return question;
   }
```

continue

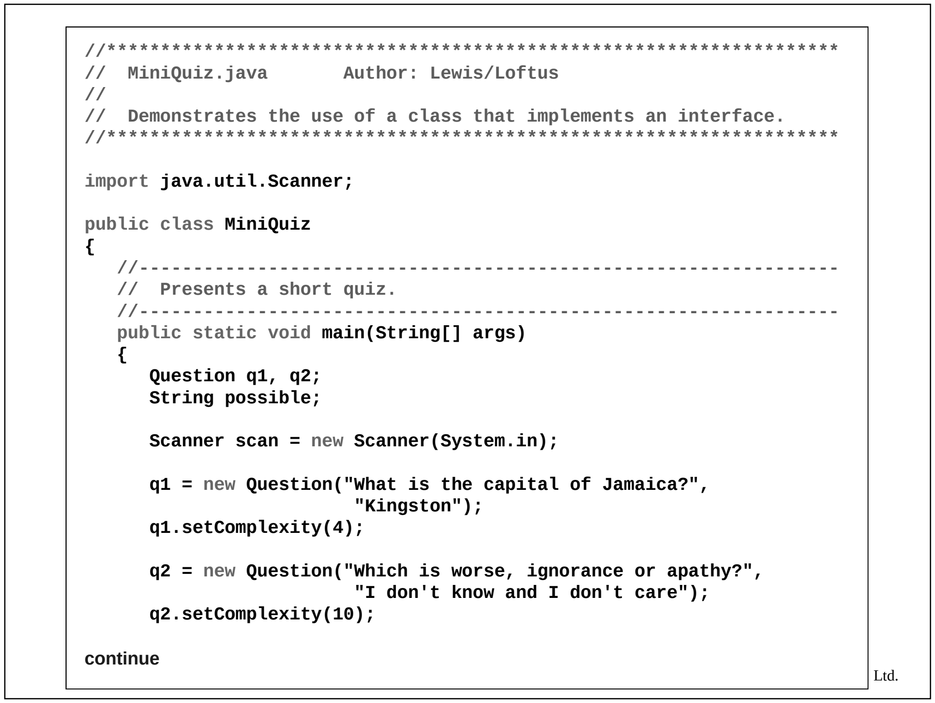


continue

```
   //-----------------------------------------------------------------
   //  Returns the answer to this question.
   //-----------------------------------------------------------------
   public String getAnswer()
   {
      return answer;
   }

   //-----------------------------------------------------------------
   //  Returns true if the candidate answer matches the answer.
   //-----------------------------------------------------------------
   public boolean answerCorrect(String candidateAnswer)
   {
      return answer.equals(candidateAnswer);
   }

   //-----------------------------------------------------------------
   //  Returns this question (and its answer) as a string.
   //-----------------------------------------------------------------
   public String toString()
   {
      return question + "\n" + answer;
   }
}
```



```
//********************************************************************
//  MiniQuiz.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a class that implements an interface.
//********************************************************************

import java.util.Scanner;

public class MiniQuiz
{
   //-----------------------------------------------------------------
   //  Presents a short quiz.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      Question q1, q2;
      String possible;

      Scanner scan = new Scanner(System.in);

      q1 = new Question("What is the capital of Jamaica?",
                        "Kingston");
      q1.setComplexity(4);

      q2 = new Question("Which is worse, ignorance or apathy?",
                        "I don't know and I don't care");
      q2.setComplexity(10);
```

continue

continue

```
      System.out.print(q1.getQuestion());
      System.out.println(" (Level: " + q1.getComplexity() + ")");
      possible = scan.nextLine();
      if (q1.answerCorrect(possible))
         System.out.println("Correct");
      else
         System.out.println("No, the answer is " + q1.getAnswer());

      System.out.println();
      System.out.print(q2.getQuestion());
      System.out.println(" (Level: " + q2.getComplexity() + ")");
      possible = scan.nextLine();
      if (q2.answerCorrect(possible))
         System.out.println("Correct");
      else
         System.out.println("No, the answer is " + q2.getAnswer());
   }
}
```

continu

Sample Run

```
What is the capital of Jamaica? (Level: 4)
Kingston
Correct

Which is worse, ignorance or apathy? (Level: 10)
apathy
No, the answer is I don't know and I don't care
```

```
                                                                   ));
      System.out.println();
      System.out.print(q2.getQuestion());
      System.out.println(" (Level: " + q2.getComplexity() + ")");
      possible = scan.nextLine();
      if (q2.answerCorrect(possible))
         System.out.println("Correct");
      else
         System.out.println("No, the answer is " + q2.getAnswer());
   }
}
```

Interfaces

- A class can implement multiple interfaces
- The interfaces are listed in the `implements` clause
- The class must implement all methods in all interfaces listed in the header

```
class ManyThings implements interface1, interface2
{
   // all methods of both interfaces
}
```