Answers to the Lab Questions

Chapter 9: Interfaces and Polymorphism

Answers :

Using an interface to share methods

Answer : 1.1

```
public interface Speakable
{
   void speak();
}

------------------------
import java.util.*;

public class AnimalRunner
{
   public static void main(String[] args)
   {
      ArrayList<Speakable> dogcatList = new ArrayList<Speakable>();
      dogcatList.add(new Dog("Fred"));
      dogcatList.add(new Cat("Wanda"));
      for (Speakable obj : dogcatList)
      {
         obj.speak();
      }
   }
}

------------------------
public class Dog implements Speakable
{
   private String name;

   public Dog(String name)
   {
      this.name = name;
   }

   public void speak()
   {
     System.out.println("Woof! Woof!");
   }

   public String toString()
   {
      return "Dog:  " + name;
   }
}

------------------------
public class Cat implements Speakable
{
   private String name;

   public Cat(String name)
   {
      this.name = name;
   }

   public void speak()
   {
     System.out.println("Meow! Meow!");
   }

   public String toString()
   {
      return "Cat:  " + name;
   }
}
```

Casting class objects

Answer : 1.2

```
public class AnimalRunner
{
   public static void main(String[] args)
   {
      Dog d1 = new Dog("Fred");
      d1.speak();
      Object obj = new Dog("Connie");
      Dog d2 = (Dog) obj;
      d2.speak();
   }
}
```

If you cast a `Cat` to a `Dog`, the compiler signals an “inconvertible types” error when the object being cast is not a member of the class which is the target of the cast.

What methods do you need to add to `BankAccount` to implement `Comparable`?

Answer : 2.1

```
int compareTo(T o)
```

Implement `compareTo` *for* `BankAccount`

2.2.

Answer : 2.2

```
/**
   Compares two bank accounts.
   @param other the other BankAccount
   @return 1 if this bank account has a greater balance than the other one,
   -1 if this bank account is has a smaller balance than the other one,
   and 0 if both bank accounts have the same balance
*/
public int compareTo(BankAccount other)
{
   if (balance > other.getBalance())
      return 1;
   else if (balance < other.getBalance())
      return -1;
   else
      return 0;
}
```

Sorting bank accounts

Answer : 2.3

```
import java.util.ArrayList;
import java.util.Collections;

public class SortTester
{
   public static void main(String[] args)
   {
      BankAccount ba1 = new BankAccount(100);
      BankAccount ba2 = new BankAccount(1000);
      BankAccount ba3 = new BankAccount(300);
      BankAccount ba4 = new BankAccount(800);
      BankAccount ba5 = new BankAccount(550);

     // Put bank accounts into a list
      ArrayList<BankAccount> list = new ArrayList<BankAccount>();
      list.add(ba1);
      list.add(ba2);
      list.add(ba3);
      list.add(ba4);
      list.add(ba5);

      // Call the library sort method
      Collections.sort(list);

      // Print out the sorted list
      for (int i = 0; i < list.size(); i++)
      {
         BankAccount b = list.get(i);
         System.out.print(b.getBalance() + " ");
      }
      System.out.println();
      System.out.println("Expected: 100 300 550 800 1000");
   }
}
```

Modifying the sorting criterion in compareTo

Answer : 2.4

```
/**
   Compares two bank accounts.
   @param other the other BankAccount
   @return -1 if this bank account has a greater balance than the other one,
   1 if this bank account is has a smaller balance than the other one,
   and 0 if both bank accounts have the same balance
*/
public int compareTo(BankAccount other)
{
   if (balance > other.getBalance())
      return -1;
   else if (balance < other.getBalance())
      return 1;
   else
      return 0;
}
```

The list is now sorted in descending order:

```
1000.0 800.0 550.0 300.0 100.0
```

Why can't you sort rectangles?

Answer : 3.1

We get an error because `Rectangle` *does not implement* `Comparable`*:*

```
CollectionsTester2.java [20:1] cannot find symbol
symbol  : method sort(java.util.ArrayList<java.awt.Rectangle>)
location: class java.util.Collections
      Collections.sort(list);
                 ^
1 error
Errors compiling SortDemo.
```

What methods are required to implement Comparator?

Answer : 3.2

```
int compare(T o1, T o2)
```

Implement a Rectangle comparator

Answer : 3.3

```
import java.util.Comparator;
import java.awt.Rectangle;

public class RectangleComparator implements Comparator<Rectangle>
{
   /**
      Compares two Rectangle objects.
      @param r1 the first rectangle
      @param r2 the second rectangle
      @return 1 if the area of the first rectangle is larger than the area of
            the second rectangle, -1 if the area of the first rectangle is
            smaller than the area of the second rectangle or 0 if the two
            rectangles have the same area
   */
   public int compare(Rectangle r1, Rectangle r2)
   {
      if (r1.getHeight() * r1.getWidth() > r2.getHeight() * r2.getWidth())
         return 1;
      else if (r1.getHeight() * r1.getWidth() < r2.getHeight() * r2.getWidth())
         return -1;
      else
         return 0;
   }
}
```

Testing the rectangle comparator

Answer : 3.4

```
import java.util.ArrayList;
import java.util.Collections;
import java.awt.Rectangle;
import java.util.Comparator;

public class RectangleSortTester
{
   public static void main(String[] args)
   {
      Rectangle rect1 = new Rectangle(5, 10, 20, 30);
      Rectangle rect2 = new Rectangle(10, 20, 30, 15);
      Rectangle rect3 = new Rectangle(20, 30, 45, 10);

      // Put the rectangles into a list
      ArrayList<Rectangle> list = new ArrayList<Rectangle>();
      list.add(rect1);
      list.add(rect2);
      list.add(rect3);

      // Call the library sort method
      Comparator<Rectangle> comp = new RectangleComparator();
      Collections.sort(list, comp);

      // Print out the sorted list
      for (int i = 0; i < list.size(); i++)
      {
         Rectangle r = (Rectangle) list.get(i);
         System.out.print(r.getWidth() + " " + r.getHeight() + " ");
      }
      System.out.println();
      System.out.println("Expected: 30 15 45 10 20 30");
   }
}
```

Making Rectangle comparator an inner class

Answer : 3.5

```
import java.util.ArrayList;
import java.util.Collections;
import java.awt.Rectangle;
import java.util.Comparator;

public class RectangleSortTester2
{
   public static void main(String[] args)
   {
      class RectangleComparator implements Comparator<Rectangle>
      {
         /**
            Compares two Rectangle objects.
            @param r1 the first rectangle
            @param r2 the second rectangle
            @return 1 if the area of the first rectangle is larger than the area of
                  the second rectangle, -1 if the area of the first rectangle is
                  smaller than the area of the second rectangle, or 0 if the two
                  rectangles have the same area
         */
         public int compare(Rectangle r1, Rectangle r2)
         {
            if (r1.getHeight() * r1.getWidth() > r2.getHeight() * r2.getWidth())
               return 1;
            else if (r1.getHeight() * r1.getWidth() < r2.getHeight() * r2.getWidth())
               return -1;
            else
               return 0;
         }
      }

      Rectangle rect1 = new Rectangle(5, 10, 20, 30);
      Rectangle rect2 = new Rectangle(10, 20, 30, 15);
      Rectangle rect3 = new Rectangle(20, 30, 45, 10);

      // Put the rectangles into a list
      ArrayList<Rectangle> list = new ArrayList<Rectangle>();
      list.add(rect1);
      list.add(rect2);
      list.add(rect3);

      // Call the library sort method
      Comparator<Rectangle> comp = new RectangleComparator();
      Collections.sort(list, comp);

      // Print out the sorted list
      for (int i = 0; i < list.size(); i++)
      {
         Rectangle r = (Rectangle) list.get(i);
         System.out.print(r.getWidth() + " " + r.getHeight() + " ");
      }
      System.out.println();
      System.out.println("Expected: 30 15 45 10 20 30");
   } }
```

Answer : 4.

```
import java.util.*;

public class Person
{
   private String name;
   private int age;
   private Memory mem;

  public Person(String name, int age)
   {
      this.name = name;
      this.age = age;
      mem = new Memory(this);
   }

   public String toString()
   {
      return "Name:  " + name + '\n' +
             "Age:  " + age + '\n';
   }

   public String getName()
   {
      return name;
   }

   public int getAge()
   {
      return age;
   }

   public void tellAll()
   {
      mem.dumpMemory();
   }

   public void rememberAnEvent(String s)
   {
       mem.addLifeData(s);
   }
}

---------------------
public class Memory
{
   ArrayList<String> lifeData;

  public Memory(Person p)
   {
      lifeData = new ArrayList<String>();
      lifeData.add("Name: " + p.getName());
      lifeData.add("Age:  " + p.getAge());
   }

   public void addLifeData(String datum)
   {
      lifeData.add(datum);
   }

   public void dumpMemory()
   {
      for (String s: lifeData)
      {
         System.out.println(s);
      }
   }
}
```