# Chapter 6: Arrays
## *Solutions*

**Multiple Choice Solutions**

1. c
2. d
3. b
4. c
5. b
6. e
7. c
8. a
9. d
10. c

**True/False Solutions**

1. T
2. F
3. F
4. T
5. T
6. F
7. F
8. T
9. T
10. T

**Short Answer Solutions**

6.1. Which of the following are valid declarations? Which instantiate an array object? Explain your answers.

- `int primes = {2, 3, 4, 5, 7, 11};`

invalid; an int cannot be declared and initialized using an intializer list; "[]" is missing

- `float elapsedTimes[] = {11.47, 12.04, 11.72, 13.88};`

  valid; the "[]" can be placed either after the element type ("float") or after the reference variable ("elapsedTimes")

- `int[] scores = int[30];`

  invalid; the right hand side of the assignment operator must contain either an initializer list or an expression of the form, "new int[30]"

- `int[] primes = new {2,3,5,7,11};`

  invalid; "new" on the right hand side of the assignment operator is neither necessary nor acceptable

- `int[] scores = new int[30];`

  valid; the assignment is correct Java syntax

- `char grades[] = {'a', 'b', 'c', 'd', 'f'};`

  valid; the assignment is correct Java syntax

- `char [] grades = new char[];`

  invalid; the size of the array must be indicated when the array is instantiated. Changing the right half of the statement to "new char[SIZE];", for example, would make the statement valid.

6.2. Describe two programs that would be difficult to implement without using arrays.

1.) a program to find the average midterm score of 600 students enrolled in an introductory computer science course

2.) a program to record and compute the sum of the snowfalls, recorded on a daily basis for the 40 days preceding the 2006 Winter Olympics

3.) a program to determine the relative frequency of each character in the Cyrillic alphabet in the original version of The Brothers Karamasov

 

4.) a program to compute the mean and standard deviation of the Dow Jones Industrial Average closings since September 11

5.) a program to store the coordinates of the vertices of polygons approximating the surface of a beating heart

6.3. Describe what problem occurs in the following code. What modifications should be made to it to eliminate the problem?

```
int[] numbers = {3, 2, 3, 6, 9, 10, 12, 32, 3, 12, 6};
for (int count = 1; count <= numbers.length; count++)
   System.out.println (numbers[count]);
```

The for loop fails to print the $0^{th}$ element of the array, and attempts to print the nonexistent $11^{th}$ element of the array. As a consequence, an ArrayIndexOutOfBoundsException is thrown. The problem can be eliminated by providing a for loop which initializes count to 0 (rather than 1) and tests if count is less than (rather than less than or equal to) numbers.length.

6.4. Write an array declaration and any necessary supporting classes to represent the following statements:

- Students' names for a class of 25 students

```
String[] students = new String[25];
```

- Students' test grades for a class of 40 students

```
int[] grades = new int[40]; // for integer percentages
or
char[] grades = new char[40]; // for simple letter grades
or
String[] grades = new String[40]; /* for letter grades with
                                     +s and -s. */
```

- Credit-card transactions that contain a transaction number, a merchant name, and a charge

```
Transactions[] charges = new Transactions[number]
/* assumes int number is assigned a value prior to the array declaration */

public class Transactions
{
      private int transactionNumber;
      private String merchantName;
      private double charge;
      .
      .
      .
}
```

- Students' names for a class and homework grades for each student

```
NameAndGrades[] theClass = new NameAndGrades [enrollees]
/* assumes int enrollees is assigned a value prior to the array declaration */

public class NameAndGrades
{
      private String name;
      private int[] grades;
      .
```

```
        .
        .
}
```

- For each employee of the L&L International Corporation: the employee number, hire date, and the amount of the last five raises

```
Employee[] LAndL = new Employee[staffSize]
/* assumes int staffSize is assigned a value prior to the array declaration */

public class Employee
{
        private int employeeNumber;
        private String hireDate;
        private double raise[] = new double[5];
        .
        .
        .
}
```

6.5. Write a method called `sumArray` that accepts an array of floating point values and returns the sum of the values stored in the array.

```
   public int sumArray (int[] values)
   {
      int sum = 0;
      for (int count = 0; count < values.length; count++)
         sum += values[count];
      return sum;
   }
```

6.6. Write a method called `switchThem` that accepts two integer arrays as parameters and switches the contents of the arrays. Take into account that the arrays may be of different sizes.

```
public void switchThem (int[] first, int[] second)
{
   if (first.length == second.length)
   {
      // copy contents of first into temp
      int [] temp = new int[first.length];
      for (int i=0; i<first.length; i++)
         temp[i] = first[i];

      //copy contents of second into first
      for (int i=0; i<first.length; i++)
         first[i] = second[i];

      //copy contents of temp into second
      for (int i=0; i<first.length; i++)
         second[i] = temp[i];
   }
   else
   {
       System.out.println("Arrays are of different "
       + "sizes and cannot be switched.")
   }
}
```

6.7. Describe a program that would use the ArrayList class instead of arrays. Describe a program that would use arrays instead of the ArrayList class. Explain your choices.

A program associated with a mail order Website for backpacks would use an object of the ArrayList class to implement the choices of colors of the backpacks because the colors and number of colors change with the seasons and as colors otherwise

gain and lose popularity. An object of the ArrayList class can grow and shrink dynamically to accommodate these changes.

A program associated with a personal day planner, with entries possible for each hour of the day, would use an array object to implement the choices for each hour of the day because the number of hours in a day, and hence the number of hours for which choices can be made for any given day, never changes. There is no need for the array object to grow or shrink to accommodate a larger or smaller number of hours.

6.8. Write a code fragment that loops through an `ArrayList<Car>` using a `foreach` loop and prints every element.

```
for (Car car : myCars)
{
   System.out.println(car);
}
```

(where myCars is the ArrayList<Car>)

6.9. Write a code fragment that loops through an `ArrayList<Car>` using a `ListIterator` and prints every element.

```
ListIterator it = myCars.listIterator();
while (it.hasNext())
{
   System.out.println(it.next());
}
```

(where myCars is the ArrayList<Car>)

6.10. Explain what would happen if the radio buttons used in the `QuoteOptions` program were not organized into a `ButtonGroup` object. Change the program to test your answer.

The three radio buttons used in the QuoteOptions program represent mutually exclusive choices (Comedy, Philosophy, or Carpentry). Their organization into a ButtonGroup prevents more than one of them from being selected at any point in time.

The references comedy, philosophy, and carpentry are associated with JRadioButtons, with comedy being set initially to true. These JRadioButtons are added to a JPanel which is the primary panel containing the GUI. If they are not organized into a ButtonGroup multiple buttons can be selected at one time. However, the quote displayed is the quote associated with the last selected button (even though other buttons also may be selected).

**AP-Style Multiple Choice Solutions**

1. E
2. A
3. C
4. D
5. C
6. B

**AP-Style Free Response Solution**

6.1

**a.**

```
public Student getValedictorian ()
{
   double bestGPA = 0;
   Student valedictorian = null;

   for (Student student : students)
   {
      if (student.getGPA() > bestGPA)
      {
         bestGPA = student.getGPA();
         valedictorian = student;
      }
   }

   return valedictorian;
}
```

**b.**

```
public double getHonorsPercent ()
{
   int numHonors = 0;

   for (Student student : students)
   {
      if (student.isHonors())
      {
         numHonors++;
      }
   }

   return (double)numHonors / students.length;
}
```

**c.**

```
private ArrayList<Student> students;
```