Short Answer

1. What is the difference between an array size declarator and a subscript?
2. Look at the following array definition.

   ```
   int[] values = new int[10];
   ```

   a. How many elements does the array have?
   b. What is the subscript of the first element in the array?
   c. What is the subscript of the last element in the array?
3. In the following array definition

   ```
   int[] values = { 4, 7, 6, 8, 2 };
   ```

   what does each of the following code segments display?

   ```
   System.out.println(values[4]);
   ```
   a. _____

   ```
   x = values[2] + values[3];
   System.out.println(x);
   ```
   b. _____

   ```
   x = ++values[1];
   System.out.println(x);
   ```
   c. _____
4. How do you define an array without providing a size declarator?
5. Assuming that `array1` and `array2` are both array reference variables, why is it not possible to assign the contents of the array referenced by `array2` to the array referenced by `array1` with the following statement?

   ```
   array1 = array2;
   ```
6. The following statement creates an `InventoryItem` array:

   ```
   InventoryItem[] items = new InventoryItem[10];
   ```

   Is it okay or not okay to execute the following statements?

   ```
   items[0].setDescription("Hammer");
   items[0].setUnits(10);
   ```
7. If a sequential search method is searching for a value that is stored in the last element of a 10,000-element array, how many elements will the search code have to examine to locate the value?
8. Look at the following array definition.

   ```
   double[][] sales = new double[8][10];
   ```

   a. How many rows does the array have?
   b. How many columns does the array have?
   c. How many elements does the array have?
   d. Write a statement that stores a number in the last column of the last row in the array.

Programming Challenges

1. `Rainfall` Class

Write a `RainFall` class that stores the total rainfall for each of 12 months into an array of `doubles`. The program should have methods that return the following:

- total rainfall for the year
- the average monthly rainfall

- the month with the most rain
- the month with the least rain

Demonstrate the class in a complete program.

Input Validation: Do not accept negative numbers for monthly rainfall figures.

2. Payroll Class

Write a `Payroll` class that uses the following arrays as fields:

- **`employeeId`**. An array of seven integers to hold employee identification numbers. The array should be initialized with the following numbers:

```
5658845 4520125 7895122 8777541
8451277 1302850 7580489
```

- **`hours`**. An array of seven integers to hold the number of hours worked by each employee
- **`payRate`**. An array of seven `doubles` to hold each employee's hourly pay rate
- **`wages`**. An array of seven `doubles` to hold each employee's gross wages

The class should relate the data in each array through the subscripts. For example, the number in element 0 of the `hours` array should be the number of hours worked by the employee whose identification number is stored in element 0 of the `employeeId` array. That same employee's pay rate should be stored in element 0 of the `payRate` array.

In addition to the appropriate accessor and mutator methods, the class should have a method that accepts an employee's identification number as an argument and returns the gross pay for that employee.

Demonstrate the class in a complete program that displays each employee number and asks the user to enter that employee's hours and pay rate. It should then display each employee's identification number and gross wages.

Input Validation: Do not accept negative values for hours or numbers less than 6.00 for pay rate.

3. Charge Account Validation

VideoNote
The Charge Account Validation Problem

Create a class with a method that accepts a charge account number as its argument. The method should determine whether the number is valid by comparing it to the following list of valid charge account numbers.

```
5658845   4520125   7895122   8777541   8451277   1302850
8080152   4562555   5552012   5050552   7825877   1250255
1005231   6545231   3852085   7576651   7881200   4581002
```

These numbers should be stored in an array. Use either a sequential search or a binary search to locate the number passed as an argument. If the number is in the array, the method should return `true`, indicating the number is valid. If the number is not in the array, the method should return `false`, indicating the number is invalid.

Write a program that tests the class by asking the user to enter a charge account number. The program should display a message indicating whether the number is valid or invalid.

4. Larger Than *n*

In a program, write a method that accepts two arguments: an array and a number *n*. Assume that the array contains integers. The method should display all of the numbers in the array that are greater than the number *n*.

5. Charge Account Modification

Modify the charge account validation class that you wrote for Programming Challenge 3 so it reads the list of valid charge account numbers from a file. Use Notepad or another text editor to create the file.

6. Driver's License Exam

The local driver's license office has asked you to write a program that grades the written portion of the driver's license exam. The exam has 20 multiple choice questions. Here are the correct answers:

1. B	6. A	11. B	16. C
2. D	7. B	12. C	17. C
3. A	8. A	13. D	18. B
4. A	9. C	14. A	19. D
5. C	10. D	15. D	20. A

A student must correctly answer 15 of the 20 questions to pass the exam.

Write a class named `DriverExam` that holds the correct answers to the exam in an array field. The class should also have an array field that holds the student's answers. The class should have the following methods:

- `passed`. Returns `true` if the student passed the exam, or `false` if the student failed
- `totalCorrect`. Returns the total number of correctly answered questions
- `totalIncorrect`. Returns the total number of incorrectly answered questions
- `questionsMissed`: An `int` array containing the question numbers of the questions that the student missed

Demonstrate the class in a complete program that asks the user to enter a student's answers, and then displays the results returned from the `DriverExam` class's methods.

Input Validation: Only accept the letters A, B, C, or D as answers.

7. Quarterly Sales Statistics

Write a program that lets the user enter four quarterly sales figures for six divisions of a company. The figures should be stored in a two-dimensional array. Once the figures are entered, the program should display the following data for each quarter:

- A list of the sales figures by division
- Each division's increase or decrease from the previous quarter (this will not be displayed for the first quarter)
- The total sales for the quarter
- The company's increase or decrease from the previous quarter (this will not be displayed for the first quarter)
- The average sales for all divisions that quarter
- The division with the highest sales for that quarter

Input Validation: Do not accept negative numbers for sales figures.

8. Grade Book

A teacher has five students who have taken four tests. The teacher uses the following grading scale to assign a letter grade to a student, based on the average of his or her four test scores.

Test Score	Letter Grade
90–100	A
80–89	B
70–79	C
60–69	D
0–59	F

Write a class that uses a `String` array (or an `ArrayList` object) to hold the five students' names, an array of five characters to hold the five students' letter grades, and five arrays of four `doubles` each to hold each student's set of test scores. The class should have methods that return a specific student's name, average test score, and a letter grade based on the average.

Demonstrate the class in a program that allows the user to enter each student's name and his or her four test scores. It should then display each student's average test score and letter grade.

Input validation: Do not accept test scores less than zero or greater than 100.

9. Grade Book Modification

Modify the grade book application in Programming Challenge 7 so it drops each student's lowest score when determining the test score averages and letter grades.

10. Lottery Application

Write a `Lottery` class that simulates a lottery. The class should have an array of five integers named `lotteryNumbers`. The constructor should generate a random number in the range of 0 through 9 for each element in the array. Refer to Chapter 4's discussion of the `Random` class for generating random numbers. The class should also have a method that accepts an array of five integers that represent a person's lottery picks. The method is to compare the corresponding elements in the two arrays and return the number of digits that match. For example, the following shows the `lotteryNumbers` array and the user's array with sample numbers stored in each. There are two matching digits (elements 2 and 4).

`lotteryNumbers` array:

User's array:

In addition, the class should have a method that returns a copy of the `lotteryNumbers` array.

Demonstrate the class in a program that asks the user to enter five numbers. The program should display the number of digits that match the randomly generated lottery numbers. If all of the digits match, display a message proclaiming the user a grand prize winner.

11. `ArrayOperations` Class

Write a class name `ArrayOperations` with the following static methods:

- `getTotal`. This method should accept a one-dimensional array as its argument and return the total of the values in the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.
- `getAverage`. This method should accept a one-dimensional array as its argument and return the average of the values in the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.
- `getHighest`. This method should accept a one-dimensional array as its argument and return the highest value in the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.
- `getLowest`. This method should accept a one-dimensional array as its argument and return the lowest value in the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.

Demonstrate the class in a complete program with test data stored in arrays of various data types.

12. Number Analysis Class

Write a class with a constructor that accepts a file name as its argument. Assume the file contains a series of numbers, each written on a separate line. The class should read the contents of the file into an array, and then display the following data:

- The lowest number in the array
- The highest number in the array
- The total of the numbers in the array
- The average of the numbers in the array

The student source code folder `Chapter 07` contains a text file named `Numbers.txt`. This file contains 12 random numbers. Write a program that tests the class by using this file.

13. Name Search

If you have downloaded this book's source code (the companion Web site is available at www.pearsonhighered.com/gaddis) you will find the following files in the Chapter 7 folder:

- GirlNames.txt—This file contains a list of the 200 most popular names given to girls born in the U.S. from the year 2000 to 2009.
- BoyNames.txt—This file contains a list of the 200 most popular names given to girls born in the U.S. from the year 2000 to 2009.

Write a program that reads the contents of the two files into two separate arrays, or `ArrayLists`. The user should be able to enter a boy's name, a girl's name, or both, and the application will display messages indicating whether the names were among the most popular.

14. Population Data

If you have downloaded this book's source code (the companion Web site is available at www.pearsonhighered.com/gaddis), you will find a file named USPopulation.txt in the Chapter 7 folder. The file contains the midyear population of the U.S., in thousands, during the years 1950 to 1990. The first line in the file contains the population for 1950, the second line contains the population for 1951, and so forth.

Write a program that reads the file's contents into an array, or an `ArrayList`. The program should display the following data:

- The average annual change in population during the time period
- The year with the greatest increase in population during the time period
- The year with the smallest increase in population during the time period

15. World Series Champions

If you have downloaded this book's source code (the companion Web site is available at www.pearsonhighered.com/gaddis), you will find a file named WorldSeriesWinners.txt. This file contains a chronological list of the World Series winning teams from 1903 to 2009. (The first line in the file is the name of the team that won in 1903, and the last line is the name of the team that won in 2009. Note that the World Series was not played in 1904 or 1994, so the file contains no entries for those years)

Write a program that lets the user enter the name of a team, and then displays the number of times that team has won the World Series in the time period from 1903 to 2009.

TIP: Read the contents of the WorldSeriesWinners.txt file into an array, or an `ArrayList`. When the user enters the name of a team, the program should step through the array or `ArrayList` counting the number of times the selected team appears.

16. `2DArrayOperations` Class

Write a class named `2DArrayOperations` with the following static methods:

- `getTotal`. This method should accept a two-dimensional array as its argument and return the total of all the values in the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.
- `getAverage`. This method should accept a two-dimensional array as its argument and return the average of all the values in the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.
- `getRowTotal`. This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The method should return the total of the values in the specified row. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.
- `getColumnTotal`. This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a column in the array. The method should return the total of the values in the specified column. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.

- `getHighestInRow`. This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The method should return the highest value in the specified row of the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.
- `getLowestInRow`. This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The method should return the lowest value in the specified row of the array. Write overloaded versions of this method that work with `int`, `float`, `double`, and `long` arrays.

Demonstrate the class in a complete program with test data stored in two-dimensional arrays of various data types.

17. Search Benchmarks

Modify the `sequentialSearch` and `binarySearch` methods presented in this chapter so they keep a count of and display on the screen the number of comparisons they make before finding the value they are searching for. Then write a program that has an array of at least 20 integers. It should call the `sequentialSearch` method to locate at least five of the values. Then it should call the `binarySearch` method to locate the same values. On average, which method makes the fewest comparisons?

18. Phone Book `ArrayList`

Write a class named `PhoneBookEntry` that has fields for a person's name and phone number. The class should have a constructor and appropriate accessor and mutator methods. Then write a program that creates at least five `PhoneBookEntry` objects and stores them in an `ArrayList`. Use a loop to display the contents of each object in the `ArrayList`.

19. Trivia Game

In this programming challenge you will create a simple trivia game for two players. The program will work like this:

- Starting with player 1, each player gets a turn at answering 5 trivia questions. (There are a total of 10 questions, 5 for each player.) When a question is displayed, four possible answers are also displayed. Only one of the answers is correct, and if the player selects the correct answer, he or she earns a point.
- After answers have been selected for all the questions, the program displays the number of points earned by each player and declares the player with the highest number of points the winner.

You are to design a `Question` class to hold the data for a trivia question. The `Question` class should have fields for the following data:

- A trivia question
- Possible answer #1
- Possible answer #2
- Possible answer #3
- Possible answer #4
- The number of the correct answer (1, 2, 3, or 4)

The `Question` class should have appropriate constructor(s), accessor, and mutator methods.

The program should create an array of 10 `Question` objects, one for each trivia question. (If you prefer, you can use an `ArrayList` instead of an array.) Make up your own trivia questions on the subject or subjects of your choice for the objects.

20. Lo Shu Magic Square

The Lo Shu Magic Square is a grid with 3 rows and 3 columns shown in Figure 7-32. The Lo Shu Magic Square has the following properties:

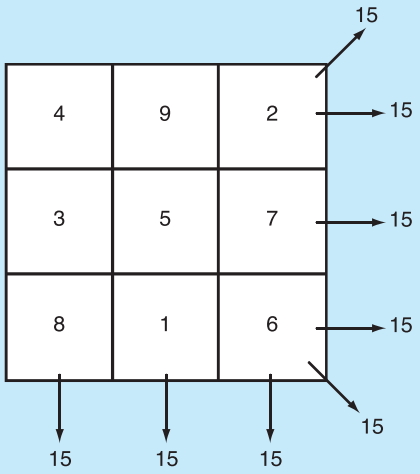
- The grid contains the numbers 1 through 9 exactly.
- The sum of each row, each column, and each diagonal all add up to the same number. This is shown in Figure 7-33.

In a program you can simulate a magic square using a two-dimensional array. Write a method that accepts a two-dimensional array as an argument and determines whether the array is a Lo Shu Magic Square. Test the method in a program.

Figure 7-32 Lo Shu Magic Square

4	9	2
3	5	7
8	1	6

Figure 7-33 Row, column, and diagonal sums in the Lo Shu Magic Square



9. What is the difference between a checked exception and an unchecked exception?
10. What is the difference between the `throw` statement and the `throws` clause?
11. What is the difference between a text file and a binary file?
12. What is the difference between a sequential access file and a random access file?
13. What happens when you serialize an object? What happens when you deserialize an object?

Programming Challenges

1. `TestScores` Class

Write a class named `TestScores`. The class constructor should accept an array of test scores as its argument. The class should have a method that returns the average of the test scores. If any test score in the array is negative or greater than 100, the class should throw an `IllegalArgumentException`. Demonstrate the class in a program.

2. `TestScores` Class Custom Exception

Write an exception class named `InvalidTestScore`. Modify the `TestScores` class you wrote in Programming Challenge 1 so it throws an `InvalidTestScore` exception if any of the test scores in the array are invalid.

3. `RetailItem` Exceptions

Programming Challenge 4 of Chapter 3 required you to write a `RetailItem` class that held data pertaining to a retail item. Write an exception class that can be instantiated and thrown when a negative number is given for the price. Write another exception class that can be instantiated and thrown when a negative number is given for the units on hand. Demonstrate the exception classes in a program.

4. `Month` Class Exceptions

Programming Challenge 5 of Chapter 6 required you to write a `Month` class that holds information about the month. Write exception classes for the following error conditions:

- A number less than 1 or greater than 12 is given for the month number.
- An invalid string is given for the name of the month.

Modify the `Month` class so it throws the appropriate exception when either of these errors occurs. Demonstrate the classes in a program.

5. `Payroll` Class Exceptions

Programming Challenge 5 of Chapter 3 required you to write a `Payroll` class that calculates an employee's payroll. Write exception classes for the following error conditions:

- An empty string is given for the employee's name.
- An invalid value is given for the employee's ID number. If you implemented this field as a string, then an empty string would be invalid. If you implemented this field as a numeric variable, then a negative number or zero would be invalid.
- An invalid number is given for the number of hours worked. This would be a negative number or a number greater than 84.

- An invalid number is given for the hourly pay rate. This would be a negative number or a number greater than 25.

Modify the `Payroll` class so it throws the appropriate exception when any of these errors occurs. Demonstrate the exception classes in a program.

6. `FileArray` Class

Design a class that has a static method named `writeArray`. The method should take two arguments: the name of a file and a reference to an `int` array. The file should be opened as a binary file, the contents of the array should be written to the file, and then the file should be closed.

Write a second method in the class named `readArray`. The method should take two arguments: the name of a file and a reference to an `int` array. The file should be opened, data should be read from the file and stored in the array, and then the file should be closed. Demonstrate both methods in a program.

7. File Encryption Filter

File encryption is the science of writing the contents of a file in a secret code. Your encryption program should work like a filter, reading the contents of one file, modifying the data into a code, and then writing the coded contents out to a second file. The second file will be a version of the first file, but written in a secret code.

Although there are complex encryption techniques, you should come up with a simple one of your own. For example, you could read the first file one character at a time, and add 10 to the character code of each character before it is written to the second file.

8. File Decryption Filter

Write a program that decrypts the file produced by the program in Programming Challenge 7. The decryption program should read the contents of the coded file, restore the data to its original state, and write it to another file.

9. `TestScores` Modification for Serialization

Modify the `TestScores` class that you created for Programming Challenge 1 to be serializable. Write a program that creates an array of at least five `TestScore` objects and serializes them. Write another program that deserializes the objects from the file.

10. Bank Account Random File

NOTE: To do this assignment, be sure to read Appendix I, available on this book's online resource page at www.pearsonhighered.com/gaddis.

One of the example classes you saw in this chapter was the `BankAccount` class. Write a `BankAccountFile` class that manages a random access file of `BankAccount` object records. The class should read and write records, and move the file pointer to any location within the file. (The class should be similar to the `InventoryItemFile` class, which is presented in Appendix I.) In a program or programs demonstrate how to create records, randomly look at records, and randomly modify records.


VideoNote
The Exception Project Problem

11. Exception Project

This assignment assumes you have completed Programming Challenge 1 of Chapter 9 (`Employee` and `ProductionWorker` Classes). Modify the `Employee` and `ProductionWorker` classes so they throw exceptions when the following errors occur:

- The `Employee` class should throw an exception named `InvalidEmployeeNumber` when it receives an employee number that is less than 0 or greater than 9999.
- The `ProductionWorker` class should throw an exception named `InvalidShift` when it receives an invalid shift.
- The `ProductionWorker` class should throw an exception named `InvalidPayRate` when it receives a negative number for the hourly pay rate.

Write a test program that demonstrates how each of these exception conditions work.