**Cis 43 Midterm => 50% of grade. Summer 2011 : Midterm Due : 12:00 PM Sunday 17 for 5 pts extra credit. For normal credit, due date is 9:20 AM Monday morning.**

**Chapter 7 Arrays and ArrayLists**

**Section 7.1**Introduction

7.1 Q1: Arrays are:

a. variable-length entities.

b. fixed-length entities.

c. data structures that contain up to 10 related data items.

d. used to draw a sequence of lines, or “rays.”’

Answer: B

**Section 7.2**Arrays

7.2 Q1: Which of the following statements about arrays are *true*?

A. An array is a group of variables containing values that all have the same type.

B. Elements are located by index or subscript.

C. The length of an array c is determined by the expression c.length();.

D. The zeroth element of array c is specified by c[ 0 ].

a. A, C, D.

b. A, B, D.

c. C, D.

d. A, B, C, D.

answer: D **B**

7.2 Q2: Consider the array:

s[ 0 ] = 7  
s[ 1 ] = 0  
s[ 2 ] = -12  
s[ 3 ] = 9  
s[ 4 ] = 10  
s[ 5 ] = 3  
s[ 6 ] = 6

The value of s[ s[ 6 ] - s[ 5 ] ] is:

a. 0.

b. 3.

c. 9.

d. 0.

Answer: C

**Section 7.3**Declaring and Creating Arrays

7.3 Q1: A programmer must do the following before using an array:

a. declare then reference the array.

b. create then declare the array.

c. create then reference the array.

d. declare then create the array.

Answer: D

7.3 Q2: Consider the code segment below. Which of the following statements is *false*?

int[] g;  
g = new int[ 23 ];

a. The first statement declares an array reference.

b. The second statement creates the array.

c. g is a reference to an array of integers.

d. The value of g[ 3 ] is -1.

Answer: D­

**Section 7.4**Examples Using Arrays

Creating and Initializing an Array

7.4 Q1: Which of the following statements about creating arrays and initializing their elements is *false*?

a. The new keyword should be used to create an array.

b. When an array is created, the number of elements must be placed in square brackets following the type of element being stored.

c. The elements of an array of integers have a value of null before they are initialized.

d. A for loop is commonly used to set the values of the elements of an array.

Answer: C

7.4 Q2: What do the following statements do?

double[] array;  
array = new double[ 14 ];

a. Create a double array containing 13 elements.

b. Create a double array containing 14 elements.

c. Create a double array containing 15 elements.

d. Declare but do not create a double array.

Answer: B

Using an Array Initializer

7.4 Q3: Which of the following initializer lists would correctly set the elements of array n?

a. int[] n = { 1, 2, 3, 4, 5 };.

b. array n[ int ] = { 1, 2, 3, 4, 5 };.

c. int n[ 5 ] = { 1; 2; 3; 4; 5 };.

d. int n = new int( 1, 2, 3, 4, 5 );.

Answer: A

Calculating a Value to Store in Each Array Element

7.4 Q4: Constant variables also are called .

a. write-only variables.

b. finals.

c. named constants.

Answer: B **C**

7.4 Q5: Which of the following will *not* produce a compiler error?

a. Changing the value of a constant after it is declared.

b. Changing the value at a given index of an array after it is created.

c. Using a final variable before it is initialized.

d. All of the above will produce compiler errors.

Answer: D **B**

Summing the Elements of an Array

7.4 Q6: Consider the program below:

public class Test  
{  
 public static void main( String[] args )  
 {  
 int[] a;  
 a = new int[ 10 ];  
  
 for ( int i = 0; i < a.length; i++ )  
 a[ i ] = i + 2;

int result = 0;  
 for ( int i = 0; i < a.length; i++ )  
 result += a[ i ];

System.out.printf( "Result is: %d\n", result );  
 } // end main  
} // end class Test

The output of this program will be:

a. Result is: 62.

b. Result is: 64.

c. Result is: 65.

d. Result is: 67.

Answer: C

7.4 Q7: Consider the class below:

public class Test  
{  
 public static void main( String[] args )  
 {  
 int[] a = { 99, 22, 11, 3, 11, 55, 44, 88, 2, -3 };  
  
 int result = 0;  
  
 for ( int i = 0; i < a.length; i++ )  
 {  
 if ( a[ i ] > 30 )  
 result += a[ i ];  
 } // end for  
  
 System.out.printf( "Result is: %d\n", result );  
 } // end main  
} // end class Test

The output of this Java program will be:

a. Result is: 280.

b. Result is: 154.

c. Result is: 286.

d. Result is: 332.

Answer: C

Using Bar Charts to Display Array Data Graphically

7.4 Q8: Which flag in a format specifier indicates that values with fewer digits than the field width should begin with a leading 0?

a. p.

b. l.

c. w.

d. 0.

Answer: D

Using the Elements of an Array as Counters

7.4 Q9: Invalid possibilities for array indices include .

a. Positive integers.

b. Negative integers.

c. Zero.

d. None of the above.

Answer: B

Using Arrays to Analyze Survey Results

7.4 Q10: Which expression adds 1 to the element of array arrayName at index i?

a. ++arrayName[ i ].

b. arrayName++[ i ].

c. arrayName[ i++ ].

d. None of the above.

Answer: A

7.4 Q11: Attempting to access an array element out of the bounds of an array, causes a(n) .

a. ArrayOutOfBoundsException.

b. ArrayElementOutOfBoundsException.

c. ArrayIndexOutOfBoundsException.

d. ArrayException.

Answer: C

7.4 Q12: Which of the following statements is *false*?

a. An exception indicates a problem that occurs while a program executes.

b. Exception handling enables you to create fault-tolerant programs that can resolve (or handle) exceptions—in many cases, this allows a program to continue executing as if no problems were encountered.

c. The catch block contains the code that might throw an exception, and the try block contains the code that handles the exception if one occurs.

d. Inside the catch block, you can use the parameter’s identifier to interact with a caught exception object.

Answer: D

**Section 7.5**Case Study: Card Shuffling and Dealing Simulation

Class ***DeckOfCards***

7.5 Q1: Consider integer array values, which contains 5 elements. Which statements successfully swap the contents of the array at index 3 and index 4?

a.

values[ 3 ] = values[ 4 ];  
values[ 4 ] = values[ 3 ];

b.

values[ 4 ] = values[ 3 ];  
values[ 3 ] = values[ 4 ];

c.

int temp = values[ 3 ];  
values[ 3 ] = values[ 4 ];  
values[ 4 ] = temp;

d.

int temp = values[ 3 ];  
values[ 3 ] = values[ 4 ];  
values[ 4 ] = values[ 3 ];

answer: C

7.5 Q2: In this question, assume a class, Book, has been defined. Which set of statements creates an array of Book objects?

a.

Book[] books;  
books = new Book[ numberElements ];

b.

Book[] books];  
books = new Book()[ numberElements ];

c.

new Book() books[];  
books = new Book[ numberElements ];

d. All of the above.

Answer: A

**Section 7.6**Enhanced for Statement

7.6 Q1: Assume array items contains the integer values 0, 2, 4, 6 and 8. Which of the following set of statements uses the enhanced for loop to display each value in array items?

a.

for ( int i = 0; i < items.length; i++ )  
 System.out.prinf( "%d\n", items[ i ] );

b.

for ( int i : items )  
 System.out.prinf( "%d\n", items[ i ] );

c.

for ( int i : items )  
 System.out.prinf( "%d\n", i );

d.

for ( int i = 0 : items.length )  
 System.out.prinf( "%d\n", items[ i ] );

answer: C

7.6 Q2: Which of the following tasks *cannot* be performed using an enhanced for loop?

a. Calculating the product of all the values in an array.

b. Displaying all even element values in an array.

c. Comparing the elements in an array to a specific value.

d. Incrementing the value stored in each element of the array.

Answer: D **C**

**Section 7.7**Passing Arrays to Methods

7.7 Q1: Which statement correctly passes the array items to method takeArray? Array items contains 10 elements.

a. takeArray( items[] ).

b. takeArray( items ).

c. takeArray( items[ 9 ] ).

d. Arrays cannot be passed to methods—each item must be sent to the method separately.

Ansswer: B

7.7 Q2: Consider array items, which contains the values 0, 2, 4, 6 and 8. If method changeArray is called with the method call changeArray( items, items[ 2 ] ), what values are stored in items after the method has finished executing?

public static void changeArray( int[] passedArray, int value )  
{  
 passedArray[ value ] = 12;  
 value = 5;  
} // end method changeArray

a. 0, 2, 5, 6, 12.

b. 0, 2, 12, 6, 8.

c. 0, 2, 4, 6, 5.

d. 0, 2, 4, 6, 12.

Answer: B

Notes on Passing Arguments to Methods

7.7 Q3: When an argument is passed by reference:

a. a copy of the argument’s value is passed to the called method.

b. changes to the argument do not affect the original variable’s value in the caller.

c. the called method can access the argument’s value in the caller directly and modify that data.

d. the original value is removed from memory.

Answer: All of the above **C**

**Section 7.8**Case Study: Class GradeBook Using an Array to Store Grades

Class ***GradeBookTest*** That Demonstrates Class ***GradeBook***

7.8 Q1: What kind of application tests a class by creating an object of that class and calling the class’s methods?

a. Pseudo application.

b. Debugger.

c. Tester.

d. Test harness.

Answer: D

**Section 7.9**Multidimensional Arrays

7.9 Q1: In Java, multidimensional arrays:

a. are not directly supported.

b. are implemented as arrays of arrays.

c. are often used to represent tables of values.

d. All of the above.

Answer: D

7.9 Q2: In array items, which expression below accesses the value at row 3 and column 4?

a. items[ 3 ].[ 4 ].

b. items[ 3[ 4 ] ].

c. items[ 3 ][ 4 ].

d. items[ 3, 4 ].

Answer: C

7.9 Q3: An array with m rows and n columns is *not*:

A. An m-by-n array.

B. An n-by-m array.

C. A two-dimensional array.

D. A dual-transcripted array.

a. A and C.

b. A and D.

c. B and D.

d. B and C.

answer: C

Arrays of One-Dimensional Arrays

7.9 Q4: Which statement below initializes array items to contain 3 rows and 2 columns?

a. int[][] items = { { 2, 4 }, { 6, 8 }, { 10, 12 } };.

b. int[][] items = { { 2, 6, 10 }, { 4, 8, 12 } };.

c. int[][] items = { 2, 4 }, { 6, 8 }, { 10, 12 };.

d. int[][] items = { 2, 6, 10 }, { 4, 8, 12 };.

Answer: A

7.9 Q5: For the array in the previous question, what is the value returned by items[ 1 ][ 0 ]?

a. 4.

b. 8.

c. 12.

d. 6.

Answer: D

Two-Dimensional Arrays with Rows of Different Lengths

7.9 Q6: Which of the following statements creates a multidimensional array with 3 rows, where the first row contains 1 element, the second row contains 4 elements and the final row contains 2 elements?

a. int[][] items = { { 1, null, null, null }, { 2, 3, 4, 5 }, { 6, 7, null, null } };.

b. int[][] items = { { 1 }, { 2, 3, 4, 5 }, { 6, 7 } };.

c. int[][] items = { { 1 }, { 2, 3, 4, 5 }, { 6, 7 }, {} );.

d. int[][] items = { { 1 }, { 4 }, { 2 } };.

Answer: B

Creating Two-Dimensional Arrays with Array-Creation Expressions

7.9 Q7: Which of the following sets of statements creates a multidimensional array with 3 rows, where the first row contains 1 value, the second row contains 4 items and the final row contains 2 items?

a.

int[][] items;  
items = new int[ 3 ][ ? ];  
items[ 0 ] = new int[ 1 ];  
items[ 1 ] = new int[ 4 ];  
items[ 2 ] = new int[ 2 ];

b.

int[][] items;  
items = new int[ 3 ][ ];  
items[ 0 ] = new int[ 1 ];  
items[ 1 ] = new int[ 4 ];  
items[ 2 ] = new int[ 2 ];

c.

int[][] items;  
items = new int[ ? ][ ? ];  
items[ 0 ] = new int[ 1 ];  
items[ 1 ] = new int[ 4 ];  
items[ 2 ] = new int[ 2 ];

d.

int[][] items;  
items[ 0 ] = new int[ 1 ];  
items[ 1 ] = new int[ 4 ];  
items[ 2 ] = new int[ 2 ];

answer: B

Two-Dimensional Array Example: Displaying Element Values

7.9 Q8: The preferred way to traverse a two-dimensional array is to use .

a. a do while statement.

b. a for statement.

c. two nested for statements.

d. three nested for statements.

Answer: C

Common Multidimensional-Array Manipulations Performed with *for* Statements

7.9 Q9: Which set of statements totals the items in each row of two-dimensional array items, and displays each total?

a.

int total = 0;  
  
for ( int row = 0; row < items.length; row++ )  
{  
 total = 0;  
  
 for ( int column = 0; column < a[ row ].length; column++ )  
 total += a[ row ][ column ];  
  
 System.out.printf( "%d\n", total );  
}

b.

int total = 0;  
  
for ( int row = 0; row < items.length; row++ )  
{  
 for ( int column = 0; column < a[ row ].length; column++ )  
 total += a[ row ][ column ];  
  
 System.out.printf( "%d\n", total );  
}

c.

int total = 0;  
  
for ( int row = 0; row < items.length; row++ )  
{  
 for ( int column = 0; column < a[ column ].length; column++ )  
 total += a[ row ][ column ];  
  
 System.out.printf( "%d\n", total );  
}

d.

int total = 0;  
  
for ( int row = 0; row < items.length; row++ )  
{  
 total = 0;  
  
 for ( int column = 0; column < a[ column ].length; column++ )  
 total += a[ row ][ column ];  
  
 System.out.printf( "%d\n", total );  
}

answer: A

Section 7.10  Case Study: Class GradeBook Using a Two-Dimensional Array

Storing Student Grades in a Two-Dimensional Array in Class ***GradeBook***

7.10 Q1: Which set of statements totals the values in two-dimensional int array items?

a.

int total = 0;  
  
for ( int subItems : items )  
 for ( int item : subItems )  
 total += item;

b.

int total = 0;  
  
for ( int item: int[] subItems : items )  
 total += item;

c.

int total = 0;  
  
for ( int[] subItems : items )  
 for ( int item : items )  
 total += item;

d.

int total = 0;  
  
for ( int[] subItems : items )  
 for ( int item : subItems )  
 total += item;

answer: D

**Section 7.11**Variable-Length Argument Lists

7.11 Q1: An argument type followed by a(n)  in a method’s parameter list indicates that the method receives a variable number of arguments of that particular type.

a. square brackets ([]).

b. ellipsis (…).

c. varargs keyword.

d. All of the above are acceptable to indicate a variable number of arguments.

Asnwer: B

**Section 7.12**Using Command-Line Arguments

7.12 Q1: Which command below runs TestProgram, and passes in the values files.txt and 3?

a. java TestProgram files.txt 3.

b. java TestProgram files.txt, 3.

c. java TestProgram "files.txt", "3".

d. java TestProgram (the arguments files.txt and 3 were passed in when the application was compiled).

Answer: A

7.12 Q2: Which method call converts the value in variable stringVariable to an integer?

a. Convert.toInt( stringVariable ).

b. Convert.parseInt( stringVariable ).

c. Integer.parseInt( stringVariable ).

d. Integer.toInt( stringVariable ).

Answer: C

Section 7.13 Class Arrays

7.13 Q1: Class Arrays methods sort, binarySearch, equals and fill are overloaded for primitive-type arrays and Object arrays. In addition, methods \_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_ are overloaded with generic versions.

1. sort, binarySearch.
2. sort, fill.
3. binarySearch, equals.
4. binarySearch, fill.

Answer: C

7.13 Q2: Class Arrays provides method \_\_\_\_\_\_\_\_\_\_ for comparing arrays.

1. compare.
2. compares.
3. equal.
4. equals.

Answer: D

Section 7.14 Introduction to Collections and Class ArrayList

7.14 Q1: Class \_\_\_\_\_\_\_\_ represents a dynamically resizable array-like data structure.

1. Array.
2. ArrayList.
3. Arrays.
4. None of the above.

Answer: B

7.14 Q1: Which of the following is false?

a. The size of an ArrayList can be determined via its length instance variable.

b. The size of an ArrayList can be determined via its size method.

c. You can add a new item to the end of an ArrayList with its add method.

c. You can get an item from a specified index in an ArrayList with its get method.

Answer: C

**Section 7.15**(Optional) GUI and Graphics Case Study: Drawing Arcs

7.13 Q1: Which method sets the background color of a JPanel?

a. setBack.

b. setBackground.

c. setBackgroundColor.

d. setColor.

Answer: B

7.13 Q2: Which of the following statements about an arc is *false*?

a. An arc is a section of an oval.

b. The sweep is the amount of arc to cover.

c. Method drawArc draws the edges of an arc.

d. The fillArc method draws an oval, with the section that is an arc filled in.

answer: A

# Chapter 8 Classes and Objects: A Deeper Look

# Section 8.2 Time Class Case Sudy

### *Time1 Class Declaration*

8.2 Q1: The \_\_\_\_\_\_\_\_\_ of a class are also called the public services or the public interface that the class provides to its clients.

1. public constructors.
2. public instance variables.
3. public methods.
4. All of the above.

Answer: C

8.2 Q2: The static method \_\_\_\_\_\_\_\_ of class String returns a formatted String.

1. printf.
2. format.
3. formatString.
4. toFormatString.

answer: B

### *Notes on the Time1 Class Declaration*

8.2 Q3: Which statement is *false*?

1. The actual data representation used within the class is of no concern to the class’s clients.
2. Clients generally care about *what* the class does but not *how* the class does it.
3. Clients are usually involved in a class’s implementation.
4. Hiding the implementation reduces the possibility that clients will become dependent on class-implementation details.

Answer: C

# Section 8.3 Controlling Access to Members

8.3 Q1: Which of the following should usually be private?

1. Methods.
2. Constructors.
3. Variables (or fields).
4. All of the above.

Answer: C

8.3 Q2: Which of the following statements is *true*?

1. Methods and instance variables can both be either public or private.
2. Information hiding is achieved by restricting access to class members via keyword public.
3. The private members of a class are directly accessible to the client of a class.
4. None of the above is true..

Answer: A

# Section 8.4: Referring to the Current Object’s Member with the this Reference

8.4 Q1: When should a program *explicitly* use the this reference?

1. Accessing a private variable.
2. Accessing a public variable.
3. Accessing a local variable.
4. Accessing a field that is shadowed by a local variable.

Answer: C

8.4 Q2: Having a this reference allows:

1. a method to refer explicitly to the instance variables and other methods of the object on which the method was called.
2. a method to refer implicitly to the instance variables and other methods of the object on which the method was called.
3. an object to reference itself.
4. All of the above.

Answer: D

# Section 8.5 Time Class Case Study: Overloaded Constructors

8.5 Q1: A constructor *cannot*:

1. be overloaded.
2. initialize variables to their defaults.
3. specify return types or return values.
4. have the same name as the class.
5. Answer: C

### *Class Time2 with Overloaded Constructors*

8.5 Q2: Constructors:

1. Initialize instance variables.
2. When overloaded, can have identical argument lists.
3. When overloaded, are selected by number, types and order of types of parameters.
4. a and c.

answer: A

### *Class Time2’s Constructors*

8.5 Q3: A programmer-defined constructor that has *no* arguments is called a(n) \_\_\_\_\_\_\_\_.

1. empty constructor.
2. no-argument constructor.
3. default constructor.
4. null constructor.

Answer: B

8.5 Q4: What happens when this is used in a constructor’s body to call another constructor of the same class if that call is not the first statement in the constructor?

1. A compilation error occurs.
2. A runtime error occurs.
3. A logic error occurs.
4. Nothing happens. The program compiles and runs.

Answer: A

### *Notes Regarding Class Time2’s* Set *and* Get *Methods and Constructors*

8.5 Q5: When implementing a method, use the class’s *set* and *get* methods to access the class’s \_\_\_\_\_\_\_\_ data.

1. public.
2. private.
3. protected.
4. All of the above.

Answer: D

# Section 8.6 Default and No-Argument Constructors

8.6 Q1: Which statement is *false*?

1. The compiler always creates a default constructor for a class.
2. If a class’s constructors all require arguments and a program attempts to call a no-argument constructor to initialize an object of the class, a compilation error occurs.
3. A constructor can be called with no arguments only if the class does not have any constructors or if the class has a public no-argument constructor.
4. None of the above.

Answer: A

# Section 8.7 Notes on *Set* and *Get* Methods

8.7 Q1: *Set* methods are also commonly called \_\_\_\_\_\_\_\_ methods and *get* methods are also commonly called \_\_\_\_\_\_\_\_ methods.

1. query, mutator.
2. accessor, mutator.
3. mutator, accessor.
4. query, accessor.

Answer: C

### *Validity Checking in* Set *Methods*

8.7 Q3: Using public *set* methods provides data integrity if:

1. The instance variables are public.
2. The instance variables are private.
3. The methods perform validity checking.
4. Both b and c.

Answer: B **D**

.

# Section 8.8 Composition

8.8 Q1: Composition is sometimes referred to as a(n) \_\_\_\_\_\_\_\_.

1. is-a relationship.
2. has-a relationship.
3. many-in-one relationship.
4. one-to-many relationship.

Answer: B

# Section 8.9 Enumerations

8.9 Q1: enum types are implicitly \_\_\_\_\_\_\_\_ and enum constants are implicitly \_\_\_\_\_\_\_\_.

1. final, static.
2. static, static.
3. static, final.
4. final, final.

Answer: C

8.9 Q2: Which statement is *false*?

1. An enum declaration is a comma-separated list of enum constants and may optionally include other components of traditional classes, such as constructors, fields and methods.
2. Any attempt to create an object of an enum type with operator new results in a compilation error.
3. An enum constructor cannot be overloaded.
4. An enum constructor can specify any number of parameters.

Answer: C

8.9 Q3: Which method returns an array of the enum’s constants?

1. values.
2. getValues.
3. constants.
4. getConstants.

Answer: A

# Section 8.10 Garbage Collection and Method finalize

8.10 Q1: Which of the following is *false*?

1. Method finalize does *not* take parameters and has return type void.
2. Memory leaks using Java are rare because of automatic garbage collection.
3. Objects are marked for garbage collection by method finalize.
4. The garbage collector reclaims unused memory.

Answer: D **C**

# Section 8.11 static Class Members

8.11 Q1: Static class variables:

1. are final.
2. are public.
3. are private.
4. are shared by all objects of a class.

Answer: D

8.11 Q2: Which of the following is *false*?

1. A static method must be used to access private static instance variables.
2. A static method has no this reference.
3. A static method can be accessed even when no objects of its class have been instantiated.
4. A static method can call instance methods directly.

Answer: C

# Section 8.12 static Import

8.12 Q1: Which syntax imports all static members of class Math?

1. static import java.lang.Math.\*.
2. import static java.lang.Math.\*.
3. static import java.lang.Math.
4. import static java.lang.Math.

answer: B

# Section 8.13 final Instance Variables

8.13 Q1: Instance variables declared final do not or cannot:

1. Cause syntax errors if used as a left-hand value.
2. Be initialized.
3. Be modified.
4. None of the above.

Answer: C

8.13 Q2: A final field should also be declared \_\_\_\_\_\_\_\_ if it is initialized in its declaration.

1. private.
2. public.
3. protected.
4. static.

Answer: D

# Section 8.14 Time Class Case Study: Creating Packages

8.16 Q1: A package is:

1. A directory structure used to organize classes and interfaces.
2. A mechanism for software reuse.
3. A group of related classes and interfaces.
4. All of the above.

Answer: D

### *Steps for Declaring a Reusable Class*

8.16 Q2: A class within a package must be declared public if

1. It will be used only by other classes in the same package.
2. It will be used by classes that are not in the same package.
3. It is in the same directory as the other classes in the package.
4. It has a unique name.

Answer: A

### *Step 3: Compiling the Packaged Class*

8.16 Q4: When compiling a class in a package, the javac command-line option \_\_\_\_\_\_\_\_ causes the javac compiler to create appropriate directories based on the class’s package declaration.

1. -p.
2. -a.
3. -d.
4. -dir.

Answer: C

### *Step 4: Importing the Reusable Class*

8.16 Q5: The import declaration import java.util.\*; is known as a \_\_\_\_\_\_\_\_.

1. single-type-import declaration.
2. all-type-import declaration.
3. multiple-import declaration.
4. type-import-on-demand declaration.

Answer: B

8.16 Q6: The import declaration import \*; \_\_\_\_\_\_\_\_.

1. causes a compilation error.
2. imports all classes in the library.
3. imports the default classes in the library.
4. imports the classes in package java.lang.

answer: B **A**

### *Specifying the Classpath During Compilation*

8.16 Q7: The classpath consists of a list of directories or archive files, each separated by a \_\_\_\_\_\_\_\_ on Windows or a \_\_\_\_\_\_\_\_ on UNIX/Linux/Max OS X.

1. colon (:), semicolon (;).
2. semicolon (;), colon (:).
3. comma (,), semicolon (;).
4. semicolon (;), comma (,).

Answer: B

8.16 Q8: By default, the classpath consists only of the \_\_\_\_\_\_\_\_. However, the classpath can be modified by providing the \_\_\_\_\_\_\_\_ option to the javac compiler.

1. root directory of the package, -d.
2. current directory, -d.
3. root directory of the package, -classpath.
4. current directory, -classpath.

Answer: D

.

# Section 8.15 Package Access

8.17 Q1: When no access modifier is specified for a method or variable, the method or variable:

1. Is public.
2. Is private.
3. Has package access.
4. Is static.

Answer: C

# Chapter 9 Object Oriented Programming: Inheritance

# Section 9.1 Introduction

9.1 Q1: Which of the following statements is *false*?

1. A subclass is often larger than its superclass.
2. A superclass object is a subclass object.
3. The class following the extends keyword in a class declaration is the direct superclass of the class being declared.
4. Java uses interfaces to provide the benefits of multiple inheritance.

Answer: B

9.1 Q2: Inheritance is also known as the

1. “knows-a” relationship.
2. “has-a” relationship.
3. “uses-a” relationship.
4. “is-a” relationship.

Answer: D

# Section 9.2 Superclasses and Subclasses

9.2 Q1: Which of the following is *not* a superclass/subclass relationship?

1. Ford/Taurus.
2. University/Boston University.
3. Sailboat/Tugboat.
4. Country/USA.

Answer: A **C**

9.2 Q2: An advantage of inheritance is that:

1. All methods can be inherited.
2. All instance variables can be uniformly accessed by subclasses and superclasses.
3. Objects of a subclass can be treated like objects of their superclass.
4. None of the above.

Answer: C

# Section 9.3 protected Members

9.3 Q1: Which of the following keywords allows a subclass to access a superclass method even when the subclass has overridden the superclass method?

1. base.
2. this.
3. public.
4. super.

Answer: D

9.3 Q2: Using the protected keyword gives a member:

1. public access.
2. package access.
3. private access.
4. block scope.

Answer: C

9.3 Q3: Superclass methods with this level of access *cannot* be called from subclasses.

1. private.
2. public.
3. protected.
4. package.

Answer: A

# Section 9.4 Relationship between Superclasses and Subclasses

# Section 9.4.1 Creating and Using a CommissionEmployee Class

9.4.1 Q1: Every class in Java, except \_\_\_\_\_\_\_\_, extends an existing class.

1. Integer.
2. Object.
3. String.
4. Class.

Answer: A

9.4.1 Q2: *Overriding* a method differs from *overloading* a method because:

1. Overloaded methods have the same signature.
2. Overridden methods have the same signature.
3. Both of the above.
4. Neither of the above.

Answer: B

# Section 9.4.2 Creating a BasePlusCommissionEmployee Class without Using Inheritance

9.4.2 Q1: To avoid duplicating code, use \_\_\_\_\_\_\_\_, rather than \_\_\_\_\_\_\_\_.

1. inheritance, the “copy-and-past” approach.
2. the “copy-and-past” approach, inheritance.
3. a class that explicitly extends Object, a class that does not extend Object.
4. a class that does not extend Object, a class that explicitly extends Object.

Answer: A

# Section 9.4.3 Creating a CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy

9.4.3 Q1: Consider the classes below, declared in the same file:

class A

{

int a;

public A()

{

a = 7;

}

}

class B extends A

{

int b;

public B()

{

b = 8;

}

}

Which of the statements below is *false*?

1. Both variables a and b are instance variables.
2. After the constructor for class B executes, the variable a will have the value 7.
3. After the constructor for class B executes, the variable b will have the value 8.
4. A reference of type A can be treated as a reference of type B.

Answer: B **D**

9.4.3 Q2: Which of the following is the superclass constructor call syntax?

1. keyword super, followed by a dot (.) .
2. keyword super, followed by a set of parentheses containing the superclass constructor arguments.
3. keyword super, followed by a dot and the superclass constructor name.
4. None of the above.

Answer: C

# Section 9.4.4 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using protected Instance Variables

9.4.4 Q1: Which superclass members are inherited by all subclasses of that superclass?

1. private instance variables and methods.
2. protected instance variables and methods.
3. private constructors.
4. protected constructors.

Answer: B

9.4.4 Q2: Which statement is *true* when a superclass has protected instance variables?

1. A subclass object can assign an invalid value to the superclass’s instance variables, thus leaving an object in an inconsistent state.
2. Subclass methods are more likely to be written so that they depend on the superclass’s data implementation.
3. We may need to modify all the subclasses of the superclass if the superclass implementation changes.
4. All of the above.

Answer: D

# Section 9.4.5 CommissionEmployee-BasePlusCommissionEmployee Inheritance Hierarchy Using private Instance Variables

9.4.5 Q1: private fields of a superclass can be accessed in a subclass

1. by calling private methods declared in the superclass.
2. by calling public or protected methods declared in the superclass.
3. directly.
4. All of the above..

Answer: B

9.4.5 Q2: Failure to prefix the superclass method name with the keyword super and a dot (.) separator when referencing the superclass’s method causes \_\_\_\_\_\_\_\_.

1. a compile-time error.
2. a syntax error.
3. infinite recursion.
4. a runtime error.

Answer: C

# Section 9.5 Constructors in Subclasses

9.5 Q1: When a subclass constructor calls its superclass constructor, what happens if the superclass’s constructor does not assign a value to an instance variable?

1. A syntax error occurs.
2. A compile-time error occurs.
3. A run-time error occurs.
4. The program compiles and runs because the instance variables are initialized to their default values.

Answer: D

# Section 9.6 Software Engineering with Inheritance

9.6 Q1: Which of the following statements is (are) *true*?

1. We can use inheritance to customize existing software.
2. A superclass specifies commonality.
3. A superclass can be modified without modifying subclasses
4. A subclass can be modified without modifying its superclass.
5. All of the above.
6. None of the above.
7. A, B and C.
8. A, B and D.

Answer: D

9.6 Q2: Which of the following is an example of a functionality that should *not* be “factored out” to a superclass?

1. Both ducks and geese are birds that know how to start flying from the water.
2. All vehicles know how to start and stop.
3. All animals lay eggs, except for mammals.
4. All paints have a color.

Answer: D **C**

# Section 9.7 Object Class

9.7 Q1: The default implementation of method clone of Object performs a \_\_\_\_\_\_\_\_.

1. empty copy.
2. deep copy.
3. full copy.
4. shallow copy.

Answer: D

9.7 Q2: The default equals implementation of class Object determines:

1. whether two references refer to the same object in memory.
2. whether two references have the same type.
3. whether two objects have the same instance variables.
4. whether two objects have the same instance variable values.

Answer: A

# Section 9.8 (Optional) GUI and Graphics Case Study: Displaying Text and Images Using Labels

9.8 Q1: Class \_\_\_\_\_\_\_\_ represents an image that can be displayed on a JLabel.

1. Image.
2. Icon.
3. ImageIcon.
4. IconImage.

Answer: C

9.8 Q2: Which method changes the text the label displays?

1. changeText.
2. setText.
3. changeLabel.
4. setLabel.

answer: B

**Chapter 10: Object-Oriented Programming: Polymorphism**

Section 10.1  Introduction

10.1 Q1: Polymorphism enables you to:

a. program in the general.

b. program in the specific.

c. absorb attributes and behavior from previous classes.

d. hide information from the user.

Answer: A

10.1 Q2: Which of the following statements about interfaces is *false*?

a. An interface describes a set of methods that can be called on an object, providing a default implementation for the methods.

b. An interface describes a set of methods that can be called on an object, not providing concrete implementation for the methods.

c. Interfaces are useful when attempting to assign common functionality to possibly unrelated classes.

d. Once a class implements an interface, all objects of that class have an is-a relationship with the interface type.

Answer: A

Section 10.2  Polymorphism Examples

10.2 Q1: For which of the following would polymorphism *not* provide a clean solution?

a. A billing program where there is a variety of client types that are billed with different fee structures.

b. A maintenance log program where data for a variety of types of machines is collected and maintenance schedules are produced for each machine based on the data collected.

c. A program to compute a 5% savings account interest for a variety of clients.

d. An IRS program that maintains information on a variety of taxpayers and determines who to audit based on criteria for classes of taxpayers.

Answer: D

10.2 Q2: Polymorphism allows for specifics to be dealt with during:

a. execution.

b. compilation.

c. programming.

d. debugging.

Answer: A

Section 10.3  Demonstrating Polymorphic Behavior

10.3 Q1: Which statement *best* describes the relationship between superclass and subclass types?

a. A subclass reference *cannot* be assigned to a superclass variable and a superclass reference *cannot* be assigned to a subclass variable.

b. A subclass reference *can* be assigned to a superclass variable and a superclass reference *can* be assigned to a subclass variable.

c. A superclass reference *can* be assigned to a subclass variable, but a subclass reference *cannot* be assigned to a superclass variable.

d. A subclass reference *can* be assigned to a superclass variable, but a superclass reference *cannot* be assigned to a subclass variable.

Answer: D

Section 10.4  Abstract Classes and Methods

10.4 Q1: A(n)  class cannot be instantiated.

a. final.

b. concrete.

c. abstract.

d. polymorphic.

Answer: C

10.4 Q2: Non-abstract classes are called:

a. real classes.

b. instance classes.

c. implementable classes.

d. concrete classes.

Answer: D

Section 10.5  Case Study: Payroll System Using Polymorphism

10.5 Q1: It is a UML convention to denote the name of an abstract class in:

a. bold.

b. italics.

c. a diamond.

d. there is no convention of the UML to denote abstract classes—they are listed just as any other class.

Answer: B

10.5 Q2: If the superclass contains only abstract method declarations, the superclass is used for:

a. implementation inheritance.

b. interface inheritance.

c. Both.

d. Neither.

Answer: A

Section 10.5.1 Abstract Superclass Employee

10.5.1 Q1: Which of the following could be used to declare abstract method method1 in abstract class Class1 (method1 returns an int and takes no arguments)?

a. public int method1();

b. public int abstract method1();

c. public abstract int method1();

d. public int nonfinal method1();

answer: C

10.5.1 Q2: Which of the following statements about abstract superclasses is *true*?

a. abstract superclasses may contain data.

b. abstract superclasses may *not* contain implementations of methods.

c. abstract superclasses must declare all methods as abstract.

d. abstract superclasses must declare *all* data members not given values as abstract.

Answer: B

Section 10.5.2  Concrete Subclass SalariedEmployee

10.5.2 Q1: Consider the abstract superclass below:

public abstract class Foo  
{  
 private int a;  
 public int b;  
  
 public Foo( int aVal, int bVal )  
 {  
 a = aVal;  
 b = bVal;  
 } // end Foo constructor  
  
 public abstract int calculate();  
} // end class Foo

Any *concrete* subclass that *extends* class Foo:

a. Must implement a method called calculate.

b. Will *not* be able to access the instance variable a.

c. Neither (a) nor (b).

d. Both (a) and (b).

answer: D

Section 10.5.5 Indirect Concrete Subclass BasePlusCommissionEmployee

10.5.5 Q1: Consider classes A, B and C, where A is an abstract superclass, B is a concrete class that inherits from A and C is a concrete class that inherits from B. Class A declares abstract method originalMethod, implemented in class B. Which of the following statements is *true* of class C?

a. Method originalMethod cannot be overridden in class C—once it has been implemented in concrete class B, it is implicitly final.

b. Method originalMethod *must be* overridden in class C, or a syntax error will occur.

c. If method originalMethod is not overridden in class C but is called by an object of class C, an error occurs.

d. None of the above.

Answer: D

Section 10.5.6 Polymorphic Processing, Operator instanceof and Downcasting

10.5.6 Q1: When a superclass variable refers to a subclass object and a method is called on that object, the proper implementation is determined at execution time. What is the process of determining the correct method to call?

a. early binding.

b. non-binding.

c. on-time binding.

d. late binding.

Answer: D

10.5.6 Q2: Every object in Java knows its own class and can access this information through method .

a. getClass.

b. getInformation.

c. objectClass.

d. objectInformation.

Answer: A

Section 10.5.7 Summary of the Allowed Assignments Between Superclass and Subclass Variables

10.5.7 Q1: Assigning a subclass reference to a superclass variable is safe:

a. because the subclass object has an object of its superclass.

b. because the subclass object is an object of its superclass.

c. only when the superclass is abstract.

d. only when the superclass is concrete.

Answer: B

Section 10.6  final Methods and Classes

10.6 Q1: Classes and methods are declared final for all but the following reasons:

a. final methods allow inlining the code.

b. final methods and classes prevent further inheritance.

c. final methods are static.

d. final methods can improve performance.

Answer: B

10.6 Q2: All of the following methods are implicitly final except:

a. a method in an abstract class.

b. a private method.

c. a method declared in a final class.

d. static method.

Answer: A

10.6 Q3: Declaring a method final means:

a. it will prepare the object for garbage collection.

b. it cannot be accessed from outside its class.

c. it cannot be overloaded.

d. it cannot be overridden.

Answer: D

Section 10.7  Case Study: Creating and Using Interfaces

10.7 Q1: An interface may contain:

a. private static data and public abstract methods.

b. only public abstract methods.

c. public static final data and public abstract methods.

d. private static data and public final methods.

Answer: C

10.7 Q2: Which of the following does *not* complete the sentence correctly?

An interface .

a. forces classes that implement it to declare all the interface methods.

b. can be used in place of an abstract class when there is no default implementation to inherit.

c. is declared in a file by itself and is saved in a file with the same name as the interface followed by the .java extension.

d. can be instantiated.

Answer: C

Section 10.7.1 Developing a Payable Hierarchy

10.7.1 Q1: The UML distinguishes an interface from other classes by placing the word “interface” in  above the interface name.

a. italics.

b. carets.

c. guillemets.

d. bold.

Answer: C

Section 10.7.2 Interface Payable

10.7.2 Q1: Interfaces can have  methods.

a. 0

b. 1

c. 2

d. any number of

answer: D

Section 10.7.3 Class Invoice

10.7.3 Q1: Which keyword is used to specify that a class will define the methods of an interface?

a. uses.

b. implements.

c. defines.

d. extends.

Answer: B

10.7.3 Q2: Which of the following is *not* possible?

a. A class that implements two interfaces.

b. A class that inherits from two classes.

c. A class that inherits from one class, and implements an interface.

d. All of the above are possible.

Answer: B

Section 10.7.4 Modifying Class Employee to Implement Interface Payable

10.7.4 Q1: A class that implements an interface but does not declare all of the interface’s methods must be declared:

a. public.

b. interface.

c. abstract.

d. final. package T1;

answer:

// Step 1: Try to run this program just as is without changing

// anything.

// Step 2: Comment out the line that has var.a so that you no longer

// will have any errors.

class FromTesterOne

{

public static void main(String args[]) {

TesterOne var = new TesterOne();

// System.out.println(" x1 " + a.x1);

//System.out.println(" a is : " + var.a);

System.out.println(" b is : " + var.b);

System.out.println(" c is : " + var.c);

System.out.println(" d is : " + var.d);

}

}