**CHAPTER 1**

**1. Which of the following symbols can be used to comment a line in Java?**

|  |  |  |
| --- | --- | --- |
|  |  | # some text |
|  |  | \* some text |
|  |  | All of these. |
|  |  | // some text |
|  |  | A double slash is used to comment out a single line. A comment can also be created using /\* some text \*/ |

**2. Which of the following statements is correct regarding Java identifiers?**

|  |  |  |
| --- | --- | --- |
|  |  | Identifier names can be made up of letters, digits, the underscore character ( \_ ), the dollar sign, and the percent sign. |
|  |  | Identifier names can be made up of letters, digits, the underscore character ( \_ ), and the dollar sign only. |
|  |  | Identifier names can be made up of letters and digits only with no special characters. |
|  |  | Identifier names can only be one of the words listed in the reserved list. |

**3. Which of the following has a correctly structured Java class:**

|  |  |  |
| --- | --- | --- |
|  |  | public static void main(String[] args)  {    public class Lincoln    {    System.out.println("Welcome to CST 200");  System.out.println("Hello world!");    }  } |
|  |  | public class Lincoln  {    public static void main(String[] args)    {    System.out.println("Welcome to CST 200");  System.out.println("Hello world!");    }  } |
|  |  | public static void main(String[] args)    {    System.out.println("Welcome to CST 200");  System.out.println("Hello world!");    } |
|  |  | public class Lincoln  {    System.out.println("Welcome to CST 200");  System.out.println("Hello world!");  } |

**4. A program is a \_\_\_\_\_\_\_ that some computer hardware executes.**

|  |  |  |
| --- | --- | --- |
|  |  | series of memory addresses |
|  |  | group of related data |
|  |  | series of instructions |
|  |  | random collection of objects |
|  |  | A software program is a series of instructions that the hardware executes one after another. |

**5. In Java, total, ToTal and TOTAL are all different identifiers.**

 True

 False

Java is case sensitive

**6. What are the four programming language levels?**

* 1. Machine code
  2. Assembly Language
  3. Compiler Language
  4. Interpreter Language

**7. What is the relationship between a class and an object? Explain in your own words.**

class is the blueprint, object takes on and uses traits from blueprint﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿﻿

**8. List and briefly explain the activities involved in the program development.**

**CHAPTER 2**

**1. The Scanner class needs to be imported, using the import statement, before it can be used in a program.**

 True

 False

Scanner is contained in the java.util package, which is not imported automatically.

**2. There is no difference between using the print and println methods.**

 True

 False

The println method prints the information sent to it, then moves to the beginning of the next line.  The print method does not advance to the next line when completed.

**3. Which of the following is not a valid string literal?**

|  |  |  |
| --- | --- | --- |
|  |  | "Java |
|  |  | "" |
|  |  | "Maybe this is a string literal." |
|  |  | "ASU" |
|  |  | "#$#\*@&^$#\*"  To be a string literal, the text must be enclosed in double quotes. A string literal can contain any valid characters, including numeric digits, punctuation and other special characters.  They can also contain no characters at all. |
|  |  |  |

**4. Which of these is an invalid assignment or declaration? (Assume there are no previous variable declarations.)**

|  |  |  |
| --- | --- | --- |
|  |  | none of the above |
|  |  | int width, meters, centimeters, millimeters; |
|  |  | int pounds = 16; oz = 1; |
|  |  | int years = 1; int months = 12; int days = 365; |
|  |  | int money, dollars = 0, quarters = 0; |

A declaration/assignment statement cannot have multiple declaration/assignments separated by semi-colons since a semi-colon denotes the end of a statement in Java.

**5. Which of the following is an invalid expression in Java?**

|  |  |  |
| --- | --- | --- |
|  |  | x = num1 / num2 % num3; |
|  |  | x = /num2 \* num1 \* num1; |
|  |  | balance = balance - withdrawn; |
|  |  | x = (num1 + 3) \* 88; |
|  |  | x = ((42 + 8) - 9); |

The invalid expression has a division without a dividend (aka 'top number').

**6. Consider the expression:**

**float result = 42 + (float)2 / 4;**

**What value is stored in ‘result’ after this line is executed?**

|  |  |  |
| --- | --- | --- |
|  |  | 42.5 |
|  |  | 11 |
|  |  | 9 |
|  |  | 42 |
|  |  | This expression will cause a compile time exception. |
|  |  | The order of operations specifies that the division is performed first.  Since 2 is cast to a float, the result of the division will be a float.  The result of 2 / 4 (which will be .5) is then added to 42 yielding 42.5. |

**7. Write a line of code that prints the following using a single call to the print method.**

Hello!

What is

the total?

**8. Write a single println statement that displays the following line.**

"123 Main Street..?" the Bellman said.

**9. Write a line of Java code that computes the average of three double type variables – d1, d2, and d3 – and stores the result in a variable of type double called avg.**

**Week 1 Questions**

**1. Which of the following are not valid Java identifiers? Why?**

#no

homework$

LONG\_\_\_\_\_\_\_\_NAME

3rdPartySoftware

#no - the hash symbol is not allowed in variable names.

3rdPartySoftware - variables may not start with a number.

**2. Why are the following valid Java identifiers not considered good identifiers?**

a

theResultOfAveragingValues

tValue

Chuck

a - not descriptive.

theResultOfAveragingValues - needs to be written more concisely.

tValue - name should document relevant information.

Chuck - not descriptive.

**3. Categorize each of the following situations as a compile-time error, run-time error, or logical error:**

calling print instead of println

dividing by zero

writing "float area = 4.14\*radius\*radius;" when computing the area of a circle.

forgetting a semicolon at the end of a programming statement

calling print instead of println - logical error.

dividing by zero - run-time error.

writing "float area = 4.14\*radius\*radius;" when computing the area of a circle - logical error.

forgetting a semicolon at the end of a programming statement - compile-time error.

**4. What is the result of the following code? Explain.**

System.out.println ("Here we go!");

System.out.println ();

System.out.print ("You could always run this in Java to double check your answer.");

System.out.println ("12345");

System.out.print ("Another.");

System.out.println ("Finally done.");

Here we go!

You could always run this in Java to double check your answer.12345

Another.Finally done.

The first line prints "Here we..." and a return. The next only prints a return. The third prints the line "You could..." but does not print a return so the fourth line prints "12345" directly after it, and followed by a return. The fifth prints "Another.", which is immediately followed by "Final..." from line six and then a return.

**5. What is wrong with the following program statement? How can it be fixed?**

System.out.println ("Some strings are very long

and don't fit on one page.");

The string literal is broken across two lines - this is not allowed. The lines can be combined or double quotes can be added to make it two strings.

System.out.println ("Some strings are very long and don't fit on one page.");

System.out.println ("Some strings are very long " + "and don't fit on one page.");

**6. What is the result of the following code? Explain.**

System.out.println ("He thrusts his fists\n\tagainst" +

" the post\nand still insists\n\the sees the \"ghost\"");

He thrusts his fists

against the post

and still insists

he sees the "ghost"

The \n characters indicates a new line character. The \t characters indicate a tab. The \" characters indicate a printed double quote.

**7. For each of the following expressions, indicate the order in which the operators will be evaluated by writing a number beneath each operator.**

(num1 % num2) / c + a

a / b + c \* d

(a – (b – c)) / d

(num1 %[1] num2) /[2] c +[3] a

a /[1] b +[3] c \*[2] d

(a –[2] (b –[1] c)) /[3] d

**Chapter 3**

### QUESTION 1

**Assume we have a variable of type String called longMessage. If we wanted to retrieve only the first five characters of this variable, which of the following lines would we use?**

|  |  |  |
| --- | --- | --- |
|  |  | String substring = longMessage.front(6); |
|  |  | String substring = longMessage.substring(0,5); |
|  |  | String substring = longMessage.firstChars(5); |
|  |  | String substring = longMessage.front(5); |
|  |  | String substring = longMessage.substring(1,5); |
|  |  |  |

String substring = longMessage.substring(0,5);

### QUESTION 2

**The String class in Java is part of which package?**

|  |  |  |
| --- | --- | --- |
|  |  | The java.lang package. |
|  |  | String is a wrapper class and not part of any package. |
|  |  | None of the given choices. |
|  |  | The java.util package. |

### The String class is part of the java.lang package.  It is not a wrapper class, and it is not part of the java.util package, therefore the other choices are wrong.

### QUESTION 3

**The automatic conversion between a primitive value and a corresponding wrapper object is known as.**

|  |  |  |
| --- | --- | --- |
|  |  | Generating |
|  |  | Autoboxing |
|  |  | Static invocation |
|  |  | Aliasing |

### QUESTION 4

**What is the name of the wrapper class for the primitive type int?**

|  |  |  |
| --- | --- | --- |
|  |  | INTEGER |
|  |  | Integer |
|  |  | IntObj |
|  |  | int has no wrapper class. |

The wrapper class for primitive data type int is Integer.

### QUESTION 5

**The Math class is part of the java.lang package.**

 True

 False

### The Math class is part of the java.lang package.  Therefore, a programmer does not need to include an explicit import statement to use its methods.

### QUESTION 6

**The System.out.printf() method is an alternative way to output information in Java.**

 True

 False

### The System.out.printf() method provides an alternative way of outputting data, but was mainly incorporated into Java to make it easier to port legacy programs written in C to the Java language.

### QUESTION 7

**The dot operator is used to access an object's methods.**

 True

 False

### QUESTION 8

**When a string literal (value) is no longer needed by a Java, the memory it uses will be freed by \_\_\_\_\_.**

|  |  |  |
| --- | --- | --- |
|  |  | None of the above. |
|  |  | garbage collection |
|  |  | restarting your computer |
|  |  | taking out the trash |

### QUESTION 9

**Would the following snippet of code work properly?**

enum Season {winter, spring, summer, fall};

Season now = Season.summer;

now = Season.summerII;

|  |  |  |
| --- | --- | --- |
|  |  | Yes, summer is a member of the enumeration. |
|  |  | No, summerII is not a member of the enumeration. |
|  |  | Yes, the syntax is all correct. |
|  |  | Yes, you can assign any value to a variable. |

### QUESTION 10

**What is the range of integers that will be generated by the following expression?**

i = generator.nextInt(15) + 5;

|  |  |  |
| --- | --- | --- |
|  |  | 5 <= i <= 19 |
|  |  | 6 <= i <= 20 |
|  |  | 0 <= i <= 15 |
|  |  | 0 <= i <= 19 |

**Chapter 4**

### QUESTION 1

**A Scanner object can use delimiters other than a space.**

 True

 False

### The delimiters used to separate tokens in a Scanner object can be explicitly set using theuseDelimiter() method.

### QUESTION 2

**A for statement always executes its loop body at least once.**

 True

 False

### A do statement always executes its loop body at least once.  A while or for statement will not if its condition evaluates to false on the first pass.

### QUESTION 3

**Java will display a compilation error if code contains an infinite loop.**

 True

 False

An infinite loop is usually caused by a logical error, and will not be caught by the compiler.

### QUESTION 4

**A logical expression can be described by a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ that lists all possible combinations of values for the variables involved in the expression.**

|  |  |  |
| --- | --- | --- |
|  |  | Switch statement |
|  |  | Equality operator |
|  |  | Nested loop |
|  |  | Truth table |

### A truth table lists out all possible combinations of values for the variables involved in an input expression and also lists the evaluation of a logical expression.

### QUESTION 5

**Consider the following Java code,**

if (count != 200)

         System.out.println("The condition was true!");

**Which of the following best describes this code snippet?**

|  |  |  |
| --- | --- | --- |
|  |  | If the variable count is close to, but not greater than 200, “The condition was true!” will be printed. |
|  |  | If the variable count is exactly equal to 200, “The condition was true!” will be printed. |
|  |  | If the variable count is exactly equal to 199 or 201, “The condition was true!” will be printed. |
|  |  | This code will not compile. |
|  |  | If the variable count is not equal to 200, “The condition was true!” will be printed. |

### QUESTION 6

**If a switch statement contains multiple cases but one case does not end with a break statement, what will happen when it is called?**

|  |  |  |
| --- | --- | --- |
|  |  | The switch will never execute. |
|  |  | The program will not compile. |
|  |  | The case without break will never be executed. |
|  |  | The switch will execute the next case statement as well. |
|  |  | It will cause an infinite loop. |

In a switch block, if the case statement does not end in a break statement, the next case will also be executed.

### QUESTION 7

**Consider the following Java code,**

name = "Apha Beta GammA";

int countA = 0;

for(int i = 0; i < name.length(); i++)

   if(name.charAt(i) == 'A')

   countA++;

System.out.println("A appears " + countA + " times");

**What will be the output of above snippet?**

|  |  |  |
| --- | --- | --- |
|  |  | "A appears 0 times" |
|  |  | "A appears 2 times" |
|  |  | "A appears 5 times" |
|  |  | This code will result in runtime Exception: IndexOutofBounds |

### QUESTION 8

**What is the Java reserved word that causes the execution of a loop to stop, and the statement following the loop to be executed?**

|  |  |  |
| --- | --- | --- |
|  |  | stop |
|  |  | halt |
|  |  | continue |
|  |  | break |

The break statement causes the execution of a loop to stop, and the statements to be subsequently executed.

The continue statement causes current iteration of a loop to stop and the condition to be evaluated again, either stopping the loop or causing the next iteration.  There are no stop or halt statements in the Java language.  A switch statement is a control structure.

### QUESTION 9

**How many times does the following loop increment the value of i?**

int i = 10;

while (i > 0)

{

    i++;

}

|  |  |  |
| --- | --- | --- |
|  |  | None |
|  |  | Infinite |
|  |  | 9 |
|  |  | 10 |

### QUESTION 10

**For what values of numOne and numTwo will the following code print *grape* more than once? Hint: can the first println even be reached?**

if(numOne > numTwo && numOne < numTwo)

System.out.println("grape");

else  {

if(numOne == numTwo) {

                System.out.println("grape");

                if(numOne > numTwo)

                      System.out.println("grape");

      }//end if numOne == numTwo

}//end else

|  |  |  |
| --- | --- | --- |
|  |  | numOne = 10  numTwo = 0 |
|  |  | None of these values. |
|  |  | numOne = 10  numTwo = 10 |
|  |  | numOne = 0  numTwo = 10 |

**Week 2**

### QUESTION 1

**Consider the following snippet of code:**

int num = 4;

int num2 = 742538931;

String place = "Mesa, Arizona";

String txt = null;

From a Java perspective, all of these variables take up the same amount of space. Why is this? Hint: are object variables treated differently than primative variables?

The num and num2 variables are storing a number (int). Regardless of the current value of the number, they take up the same amount of space since Java must reserve enough memory to store any number that can be represented by an int.

Strings are objects, and as such, their variables store the addresses of the strings in memory, not the strings themselves. Since addresses are locations in the main memory, Java is simply storing the number of that location. The string data is located in the main memory and takes up space that corresponds to the size of the string.

### QUESTION 2

**Assume you have the following code:**

**String original = "Core Data Structures";**

**String change;**

Write an assignment for change that initializes it to the characters stored in  original with all 'e' characters changed to 'j'.

change = original.replace('e','j');

### QUESTION 3

**Consider the following code:**

String s1 = "Programming in Java is fun!"

String s2 = s1.toUpperCase();

String s3 = s2;

**How many String objects (not variables) does this code create? What if the last line read String s3 = s1.toLowerCase();?**

Two, one for "Programing in Java is fun!" and one for the upper case version of it. If the last line was changed, then there would be three objects with the last being an lower case version of the s1 string.

### QUESTION 4

**Write a series of expressions using the nextInt method to generate random numbers in the following specified ranges, including both endpoints. Use the version of the nextInt method that accepts a single integer parameter.**

* + 5 to 25
  + -10 to 0

int num1 = random.nextInt(21)+5;

int num2 = random.nextInt(11)-10;

**QUESTION 5**

**Write two lines of code that prompt the user for a double value, and then print the result of raising that value to the third power.**

double value = scan.nextDouble();

System.out.print(Math.pow(value, 3));

### QUESTION 6

**Write a declaration for an enumerated type that represents the days of the week.**

enum DayOfWeek { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday }

### QUESTION 7

**What is wrong with the following code fragment? Will this code compile if it is part of an otherwise valid program?**

if (form\_completed = true)

System.out.println ("The form has been completed.");

The condition in the if is an assignment (a single equal), not a conditional (two equals). The Java compile will give a type warning.

### QUESTION 8

**Imagine you have a program calculating average temperate over the day. The program stores the sum of all measured temperates in a variable called sum, and the number of measurements in count. A snippet to warn the user when the average tempurate is over 90 is given below.**

if(((float)sum / count) > 90 && count > 0)

  System.out.println("Warning: the average tempurate is over 90.");

**What happens if this code is executed before a tempurature reading has been taken?**

**Can the code be slightly altered to prevent this behavior from happening?**

There is a possible a divide by zero error. Note that this code may execute fine in Java, as the sum is a float, which causes count to be presented as a float. Since computers have imprecise storaging for floating numbers, count may have a value like 0.0000000001 which allows division. On other platforms, is it actually possible to store an exact zero. This code can be made more safe by breaking the if into nested ifs, or the two terms can be swapped:

if(count > 0 && ((float)sum / count) > 90)

### QUESTION 9

**Convert the while loop below into an equivalent for loop. (It needs to produce the same output.)**

int num = -10;

while (num < 0)

{

num++;

System.out.println (num);

}

for(int i = -9; i <0+1; i++) {

    System.out.println(i);

}

### QUESTION 10

**Consider the following code:**

int fillvalue = 0;

for(int i = 1; i < 10; i++)

{

for(int j = 1; j < 10; j++)

{

System.out.print(fillvalue);

}

System.out.print("\n");

}

**How many times will the number 0 be printed?**

It will print 81 times

**CHAPTER 5**

### QUESTION 1

**What is the name of the diagram used to show the content of, and relationships between, classes?**

|  |  |  |
| --- | --- | --- |
|  |  | class and object diagram |
|  |  | UML diagram |
|  |  | method diagram |
|  |  | XML diagram  A UML diagram helps us visualize the contents and relationships among the classes of a program.  The other choices do not refer to any type of diagram. |

### QUESTION 2

**When a class contains multiple definitions of a method, the method is called \_\_\_\_\_\_\_\_\_.**

|  |  |  |
| --- | --- | --- |
|  |  | none of the given choice is correct. |
|  |  | overridden |
|  |  | overclocked |
|  |  | overloaded |

### A method that has multiple definitions is an overloaded method.  The versions of an overloaded method are distinguished by the number, type and order of their parameters.  Overridden methods are methods that have been redefined later in an inheritance hierarchy. The choices overlocked and overlooked are not types of methods in Java.

### QUESTION 3

**All methods (except constructors) must have a return type.  What should be the return type for a method that does not return a value?**

|  |  |  |
| --- | --- | --- |
|  |  | String |
|  |  | double |
|  |  | void |
|  |  | public |

### Methods that do not need to return any data should have void specified as the return type.  A method cannot have public specified as its return type, so choice public is incorrect.  Choice int and choice double specify a return type, and therefore they must return data of that type.

### QUESTION 4

**A \_\_\_\_\_ is a step-by-step process for solving a problem.**

|  |  |  |
| --- | --- | --- |
|  |  | class |
|  |  | algorithm |
|  |  | UML diagram |
|  |  | aggregate object |

### An algorithm is a step-by-step solution for solving a problem.  A UML diagram is a way of visually representing how classes and objects interact.  An aggregate object is an object that is composed, in part, of other objects.  A class can be thought of as a blueprint for a set of objects.

### 

### QUESTION 5

**Which object design principle refers to the idea that an object should not allow its data to be accessed by unrelated parts of the program?**

|  |  |  |
| --- | --- | --- |
|  |  | encapsulation |
|  |  | methods |
|  |  | inheritance |
|  |  | instance variables |

Encapsulation is the object-oriented principle that specifies that an objects data should be guarded from inappropriate access.  Therefore choice Encapsulation is correct.  Inheritance and polymorphism are features of object-oriented programming that allow for class flexibility and re-use.  Instance variables and methods play important roles in object-oriented programming, but are not fundamental principles.

### QUESTION 6

**Which of the following method headers is most likely a header for a mutator method?**

|  |  |  |
| --- | --- | --- |
|  |  | none of these are headers for a mutator method |
|  |  | public void setSize(int size) |
|  |  | public int getSize() |
|  |  | public double computeArea() |

Mutators are methods that change the value of an instance variable, and are often referred to as “setters.”  Therefore, the choice public void setAge(int newAge) is the correct answer.

The choice public int getAge() is an example of a header for an accessor method, often referred to as a “getter.”

The choice public Person() is a constructor, and choice public double computeSalary() is a class method.

### QUESTION 7

**Code in the main method can only access local or static variables.**

 True

 False

### 

### A main method cannot access non-static and non-local variables because it is a static method.  In particular, it cannot access any variables declared at the class level.

### QUESTION 8

**A constructor must be implemented for every class.**

 True

 False

Every class automatically has a default constructor that doesn't take any parameters

**CHAPTER 7**

### QUESTION 1

**Consider the following Java code snippet,**

**final** **int** LIMIT = 5, MULTIPLE = 10;

**int**[] list = **new** **int**[LIMIT];

**for** (**int** index = 0; index <= LIMIT; index++){

        list[index] = index \* MULTIPLE;

**for** (**int** value : list)

**System**.out.print(value + "  ");

}

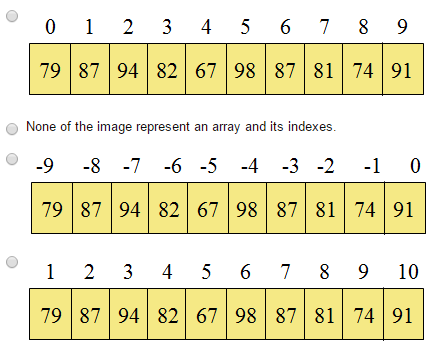
**What will be the output of above code?**

|  |  |  |
| --- | --- | --- |
|  |  | The code will end in run-time error: **IndexOutOfBounds** |
|  |  | None of the given choices is correct. |
|  |  | 0 10 20 30 40 |
|  |  | The code will end in compile-time error:**syntax error** |

### QUESTION 2

**Which image best illustrates the indices of the following array?**

int[] numbers = {79, 87, 94, 82, 67, 98, 87, 81, 74, 91};



ANSWER: A

### QUESTION 3

**Does Java support storing multiple types of primative data in the same array? E.g., half doubles, half booleans.**

|  |  |  |
| --- | --- | --- |
|  |  | Yes, each index of an array can be assigned a value of any type. |
|  |  | None of these answers. |
|  |  | No, arrays can't store booleans. |
|  |  | No, an array has a single type in its declaration. |

 However, arrays of objects can provide similar functionality.

### QUESTION 4

**Which of the statements is true about the following code snippet?**

   int[] array = new int[34];

   array[34] = 7;

|  |  |  |
| --- | --- | --- |
|  |  | The integer value 34 will be assigned to the 7th index in the array. |
|  |  | The integer value 7 will be assigned to the first value in the array. |
|  |  | The integer value 7 will be assigned to the last index in the array. |
|  |  | This code snippet will result in a run-time error. |

This code will throw an ArrayIndexOutOfBoundsException, since the last index in this array will be 33.  This causes a run-time error.

As there is no error in the syntax of the code, so there is no compile-time error.

### QUESTION 5

**In Java, the array itself is  \_\_\_\_\_\_\_ .**

|  |  |  |
| --- | --- | --- |
|  |  | an object |
|  |  | a static variable |
|  |  | a variable |
|  |  | a reserved word. |

In Java, the array itself is an object that must be instantiated.

### QUESTION 6

**Assume we have an array of String objects called names.  Which of the following for loops will**not**correctly process each element in the array.**

|  |  |  |
| --- | --- | --- |
|  |  | for(String name : names) |
|  |  | for(int j = 0; j < names.length(); j++) |
|  |  | all of these will correctly process each element |
|  |  | for(int i = 0; i < names.length; i++) |

### QUESTION 7

**\_\_\_\_\_\_\_ are arrays that contain more than one dimension.**

|  |  |  |
| --- | --- | --- |
|  |  | Multidimensional arrays |
|  |  | Overloaded arrays |
|  |  | Static arrays |
|  |  | Ragged arrays |

Multidimensional arrays are arrays that contain more than one dimension.

### QUESTION 8

**If a program attempts to access an element outside of the range of the array indexes, a run-time error will occur.**

 True

 False

If a program attempts to access an element outside of the range of the array indexes, an *ArrayOutOfBoundsException* will be thrown at run-time.

**Chapter 8**

### QUESTION 1

**If you declare a class as final, then it \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.**

|  |  |  |
| --- | --- | --- |
|  |  | cannot be used in a program. |
|  |  | cannot have subclasses. |
|  |  | cannot have superclasses. |
|  |  | has several abstract methods. |

The final modifier restricts inheritance.  In particular, a class declared as final cannot have subclasses.

### QUESTION 2

**If you need to create default a generic class, containing complete method definitions as well as methods lacking a body of code, what would be the technical term for the class?**

|  |  |  |
| --- | --- | --- |
|  |  | generic |
|  |  | super |
|  |  | interface |
|  |  | abstract |

### An abstract class represents a generic entity that is not completely defined.  An abstract class cannot be instantiated.  It contains one or more abstract methods, which are methods that should be overridden by subclasses.

### QUESTION 3

**In Java, all classes are children of what class?**

|  |  |  |
| --- | --- | --- |
|  |  | Default |
|  |  | String |
|  |  | Object |
|  |  | Class |

### QUESTION 4

**If a subclass's constructor does not include an explicit call to a superclass's constructor, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.**

|  |  |  |
| --- | --- | --- |
|  |  | none of the given options is correct. |
|  |  | a compile-time error will result. |
|  |  | the class will be implicitly declared as abstract. |
|  |  | the constructor will be called anyway. |

### The child's constructor will implicitly call the superclass's constructor if it is not done explicitly.  This will ensure that the class is properly initialized.

### QUESTION 5

**In order to prevent derived classes, as well as other classes, from accessing data members and methods of a class, the data members and methods should be declared using the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ modifier.**

|  |  |  |
| --- | --- | --- |
|  |  | static |
|  |  | public |
|  |  | private |
|  |  | protected |

Data members and methods declared using the protected modifier can be accessed by subclasses in an inheritance hierarchy, but are still encapsulated from classes and methods outside of the hierarchy.

### QUESTION 6

**Consider the following concepts. Which of them would most likely need to created as an abstract class?**

|  |  |  |
| --- | --- | --- |
|  |  | Koala |
|  |  | Cat |
|  |  | Animal |
|  |  | Squid |

The *Animal* class is most likely to be abstract since it is the most generic.

### QUESTION 7

**The derived class created using inheritance is called ?**

|  |  |  |
| --- | --- | --- |
|  |  | a child class |
|  |  | super class |
|  |  | sister class |
|  |  | parent class |

### QUESTION 8

**Assume we have some class A, and another class B which is derived (inherits) from A. We say that A and B have a \_\_\_\_\_\_\_ relationship.**

|  |  |  |
| --- | --- | --- |
|  |  | is-a |
|  |  | has-a |
|  |  | cordial |
|  |  | static |

### Inheritance should establish an *is-a* relationship.  Therefore any objects that are of a type lower in the inheritance hierarchy are also of a type higher in the inheritance hierarchy.

### QUESTION 9

**A child class cannot define a method with the same name and parameter list as a method in the parent class.**

 True

 False

A subclass is allowed to override methods that are in the parent class.

### QUESTION 10

**A subclass can extend any number of parent classes at once.**

 True

 False

Allowing a subclass to extend multiple parent classes leads to multiple inheritance, which is not supported in Java.

**WEEK 3**

### QUESTION 1

**For each of the following pairs, which represents a class and which represents an object of that class?**

* 1. Course, CST200
  2. Calvin, Person
  3. USSArizona, Ship
  4. Hulk, Superhero
  5. Magazine, Economist
  6. Thanksgiving, Holiday

1. Course is a class and CST200 is an object.
2. Person is a class and Calvin is an object.
3. Ship is a class and USSArizona is an object.
4. Superhero is a class and Hulk is an object.
5. Magazine is a class and Economist is an object.
6. Holiday is a class and Thanksgiving is an object.

### QUESTION 2

**Complete the following method called repeatString (outline below), which returns a String where the String input parameter (element) has been repeated a number of times. For example, if the parameter values are "stat" and 4, the return value is"statstatstatstat". If times is less than 2, return element directly.**

String repeatString(String element, int times)

{

    //Implement this.

}

String repeatString(String element, int times)

{

  if(times < 2)

    return element;

  else

  {

    String dupe = "";

    for(int i = 0; i < times; i++)

    dupe += element

    return dupe;

  }

}

### QUESTION 3

**Overload the repeatString method from Question 2 such that if the integer parameter is not provided, the method acts as if times was set to 3. For example, if the parameter is "cat", the return value is "catcatcat".**

String repeatString(String element)

{

  return repeatString(String element, 3);

}

### QUESTION 4

**Create array declarations for the following descriptions of arrays:**

* 1. students' names for a class of 150 students
  2. students' GPAs for a class of 40 students
  3. last week's temperatures: 99 103 106 107 109 110 109
  4. for each employee of the L&L International Corporation: the employee number, hire date, and the amount of the last five raises. Hint: create a (simple) object containing the different pieces of information, then use it to declare an array of objects.

String[] studentNames = new String[150];

double[] studentGPA = new double[40];

int[] temperatures = {99, 103, 106, 107, 109, 110, 109};

class Employee {

  public int number;

  public String hireDate;

  public double[] raises = new double[5];

}

Employee[] llIntlCorp;

QUESTION 5

**Does the following code execute properly? If not, what should be changed?**

int[] numbers = {4, 2, 1, 6, 9, 56, 12, 4, 3, 12, 6};

//print out each number in the array.

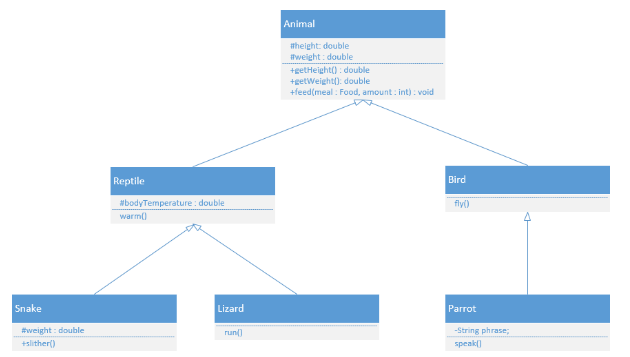
for (int i = 1; i <= numbers.length; i++)

    System.out.println(numbers[i]);

It does not execute properly, it is printing indices 1 to 11 while the array contains indices 0 to 10. The loop should be changed to "for (int i = 0; i < numbers.length; i++)".

### QUESTION 6

**Read over the following UML diagram and answer the following questions:**



* 1. What is the parent class of Lizard?
  2. Are there any shadow variables? If so, which?
  3. Does the diagram include any overloaded methods? If so, which?
  4. Can the Lizard class access the weight variable defined in Animal? Explain in a sentence.

|  |
| --- |
|  |
| * Reptile is the parent class of Lizard. * The weight variable in Snake shadows the weight variable in Animal. * There are no overloaded methods. * Yes, it can access weight directly (since it is protected, not private), or it can call getWeight(). |

### QUESTION 7

**Organize the following classes into a hierarchy: Triangle, Polygon, Point, Shape, Rectangle, and Square. Indicate (and explain) if any of these classes should be declared as abstract.**

Use an indented bullet list to format your hierachy. For a given class, indicate its children by indenting them. An example is given below:

* + Animal
    - Reptile
      * Snake
      * Lizard
    - Bird
      * Parrot

(There are multiple approaches.)

* Point (or can be under Shape)
* Shape (Should be abstract - the *idea* of a shape doesn't define enough information to compute concrete values like area.)
  + Polygon
    - Triangle
    - Rectangle
      * Square

### QUESTION 8

**Review the UML diagram from Question 6 and answer the following questions:**

* 1. If you needed to add a new class called Chicken, what would be the best parent class for it? Explain.
  2. Is Reptile a good place to put the bodyTemperature variable? Explain.
  3. Assume that the fly() method in Bird is a full implementation (not abstract). If you needed to add a child class to Bird called Ostrich, what could you do in Ostrich to disable flight? Explain.
* Bird would be the best parent class for Chicken. It makes sense to say a Chicken is-a Bird (inheritance) since a chicken has the features (e.g. wings, feathers) that birds have.
* Not really. While body temperature is very important for reptiles, it is a property of all animals. It should be higher in the inheritance tree hierarchy, so, for example, we can track individual temperatures in case an animal is ill.
* We could add a new fly() method in Ostrich that doesn't do anything. This would override method/functionality in Bird so that Ostriches cannot fly. It also might be a good idea to think about making fly() abstract or implementing flying in another way (component design).

**Midterm Study Guide**

### Question 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | **A syntax error is a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |  |  |
| |  |  | | --- | --- | | Selected Answer: | Correct  a compile-time error |  |  |  | | --- | --- | | Response Feedback: | A program that contains a syntax error is invalid, and therefore cannot be compiled.  It is a compile-time error because it is caught by the compiler.  A logical error is an error that causes a running program to behave in an unexpected manner during run-time.  A bug is an example of a logical error.  A run-time error is an error that happens while the program is running.  In Java, run-time errors are called exceptions. | |  |  |  |

### Question 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | **Consider the following two different snippets of Java code**  **//snippet 1**  **public static void main(String [] args)  {**  **System.out.println("Hi!");**  **}**  **//snippet 2**  **public static void main(String [] args){System.out.println("Hi!");}**  **Are these two snippets of code are identical from the point of view of the compiler?** |  |  |  |
| |  |  | | --- | --- | | Selected Answer: | Correct  Yes,  from the point of view of the compiler the snippets are identical. |  |  |  | | --- | --- | | Response Feedback: | These snippets of code are identical because Java ignores white space. | |  |  | |  |  |  |

### Question 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | **Consider the following Java code snippet,**  String hello = new String("Hello World!");  hello = hello.replace('H', 'W');  hello = hello.replace('W', 'H');  System.out.println(hello);    What will be the output? |  |  |  |
| |  |  | | --- | --- | | Selected Answer: | Correct  "Hello Horld!" | |  | This code will result in runtime error: Cannot replace String | |  |  |  |

### Question 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A Scanner object can use delimiters other than a space.** |  |  |  |
| |  |  | | --- | --- | | Selected Answer: | Correct True |  |  |  | | --- | --- | | Response Feedback: | The delimiters used to separate tokens in a Scanner object can be explicitly set using the useDelimiter() method. | |  |  |  |

### Question 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Suppose we have an array of String objects identified by the variable names.  Which of  the following for loops will**not**correctly process each element in the array** |  |  |  |
| |  |  | | --- | --- | | Selected Answer: | Correctc.  for(int i = 0; i < names.length(); i++) |  |  |  | | --- | --- | | Response Feedback: | Choice c will not process each element correctly due to a syntax error.  The length variable is not a method and, therefore, does not have parenthesis after it.  Choice b is an example of using a foreach loop to process an array, and choice a is a correct for loop. | |  |  |  |

### Question 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | **In Java a subclass can only *extend* one parent class.** |  |  |  |
| |  |  | | --- | --- | | Selected Answer: | Correct True |  |  |  | | --- | --- | | Response Feedback: | Allowing a subclass to extend multiple parent classes leads to multiple inheritance, which is not supported in Java. | |  |  |  |