Name: ______________________________________

1

Consider the following class declaration.

```
public class Computer
{
    private String operatingSystem;

    public Computer()
    {
        operatingSystem = "MacOS";
    }

    public Computer(String os)
    {
        operatingSystem = os;
    }
}
```

Which of the following code segments, when appearing in a class other than **Computer**, would create a reference of type **Computer** with a value of `null` ?

- **A.** **`Computer myLaptop = new Computer();`**
- **B.** **`Computer myLaptop;`**
- **C.** **`Computer("Windows");`**
- **D.** **`Computer myLaptop = new Computer("Linux");`**
- **E.** **`Computer myLaptop = new Computer("");`**

2

Consider the following class declaration.

```
public class AssetRecord
{
    private String description;
    private String dateCode;
    private double pricePrice;
    private boolean canBeMoved;

    public AssetRecord(String descriptionP, String dateCodeP, boolean canBeMovedP)
    {
        description = descriptionP;
        dateCode = dateCodeP;
        canBeMoved = canBeMovedP;
    }

    public AssetRecord(String descriptionP, String dateCodeP)
    {
        description = descriptionP;
        dateCode = dateCodeP;
        canBeMoved = true;
    }
}
```

A new constructor is to be added to the AssetRecord class. Which of the following is **NOT** a possible header for the new constructor?

- **A.** **`AssetRecord()`**
- **B.** **`AssetRecord(String descriptionP)`**
- **C.** **`AssetRecord(String descriptionP, boolean canBeMovedP)`**
- **D.** **`AssetRecord(boolean canBeMovedP)`**
- **E.** **`AssetRecord(String descriptionP, String dateCodeP, boolean canBeMovedP)`**

3

Consider the following Vcylinder class.

```
public class Vcylinder
{
    private int radius;
    private int height;

    public Vcylinder(int w, int h)
    {
        radius = w;
        height = h;
    }

    public Vcylinder(int len)
    {
        radius = len;
        height = len;
    }
}
```

Which of the following declarations, appearing in a method in a class other than Vcylinder, will correctly instantiate a Vcylinder object?

I. `Vcylinder b1 = new Vcylinder(4);`
II. `Vcylinder b2 = new Vcylinder(2, 8);`
III. `Vcylinder b3 = new Vcylinder(4.0, 4.0);`

- **A. I only**
- **B. II and III only**
- **C. I and II only**
- **D. II only**
- **E. I and III only**

4

Consider the following method, which returns the greater of its two parameters.

```
public int max(int first, int second)
{  /* implementation not shown */  }
```

Assume that each of the following expressions appears in a method in the same class as the method **max**. Assume also that the int variables **p**, **q**, and **r** have been properly declared and initialized. Which of the following expressions evaluates to the maximum value among **p**, **q**, and **r**?

I. `max(p, q), r)`
II. `max(p, max(q, r))`
III. `max(p, q), p)`

- **A. I only**
- **B. II only**
- **C. III only**
- **D. I and II only**
- **E. I, II, and III**

5

Consider the following expression.

```
(3 + 4 == 7) != (3 + 4 >= 8)
```

What value, if any, does the expression evaluate to?

- **A. true**
- **B. false**
- **C. 5**
- **D. 7**
- **E. No value; relational operators cannot be used on arithmetic expressions.**

6

Consider the following code segment, which uses properly declared and initialized int variables x and y and the String variable result.

```
String result = "";
if (x < 5)
{
    if (y > 0)
    {
        result += "a";
    }
    else
    {
        result += "b";
    }
}
else if (x > 10)
{
    if (y < 0)
    {
        result += "c";
    }
    else if (y < 10)
    {
        result += "d";
    }
    result += "e";
}
result += "f";
```

What is the value of result after the code segment is executed if x has the value **15** and y has the value **-5** ?

- **A. ad**
- **B. adf**
- **C. d**
- **D. def**
- **E. cef**

7

Consider the following code segment, which is intended to store the sum of multiples of 2 between 2 and 100, inclusive (2 + 4 + ... + 100), in the variable **total**.

```
int x = 100;
int total = 0;
while( /* missing code */ )
{
    total = total + x;
    x = x - 2;
}
```

Which of the following can be used as a replacement for `/* missing code */` so that the code segment works as intended?

- **A. `x < 100`**
- **B. `x <= 100`**
- **C. `x > 2`**
- **D. `x >= 2`**
- **E. `x != 2`**

8

Consider the following method.

```
public String mystery(String word, int num)
{
    String result = "";
    for (int k = num; k >= 0; k--)
    {
        result += word.substring(0, k);
    }
    return result;
}
```

What is returned as a result of the call `mystery("computer", 4)` ?

- **A. ccocom**
- **B. comcoc**
- **C. ccocomcomp**
- **D. compcomcoc**
- **E. comcomcomcom**

9

Consider the following code segment.

```
int a = 1;
while (a <= 2)
{
    int c = 1;
    while (/* missing condition */)
    {
        System.out.print("#");
        c++;
    }
    a++;
}
```

The code segment is intended to print "####". Which of the following can be used to replace `/* missing condition */` so that the code segment works as intended?

- **A. c <= 2**
- **B. c < 3**
- **C. c <= 3**
- **D. c > 2**
- **E. c >= 3**

10

Consider the following code segment.

```
for (int j = 0; j < 3; j++)
{
    for (int k = 0; k < j; k++)
    {
        System.out.println("hello");
    }
}
```

Which of the following best explains how changing the inner for loop header to `for (int k = j; k < 3; k++)` will affect the output of the code segment?

- **A. The output of the code segment will be unchanged.**
- **B. The string "hello" will be printed two additional times because the inner loop will iterate one additional time for each iteration of the outer loop.**
- **C. The string "hello" will be printed three additional times because the inner loop will iterate one additional time for each iteration of the outer loop.**
- **D. The string "hello" will be printed two fewer times because the inner loop will iterate one fewer time for each iteration of the outer loop.**
- **E. The string "hello" will be printed three fewer times because the inner loop will iterate one fewer time for each iteration of the outer loop.**

11

Consider the following code segment.

```
int a = 16;
while (a > 1)
{
    System.out.println("我");
    a /= 2;
}
```

How many times is **我** printed as a result of executing the code segment?

- **A. 0**
- **B. 4**
- **C. 6**
- **D. 7**
- **E. 50**

12

Consider the following class.

```
public class Item
{
    private static String description;
    private double price;
    private static double taxPercentage;

    public Item(String d, double p, double t)
    {
        description = d;
        price = p;
        taxPercentage = t;
    }

    public double totalAmount()
    {
        return price + calculateTax(price);
    }

    public static double calculateTax (double p) {
        return p * taxPercentage;
    }
}
```

Assume that an **Item** object **laptop** has been declared and initialized in another class:

```
Item laptop = new Item("MacBook", 7000.00, 0.1);
```

Which of the following code segments will successfully print the **total** price amount associated with **laptop**?

A. **`System.out.print( Item.calculateTax() );`**
B. **`System.out.print( Item.calculateTax( 7000.0 ) );`**
C. **`System.out.print( Item.totalAmount() );`**
D. **`System.out.println( laptop.totalAmount() );`**
E. **`laptop.totalAmount();`**

13

The **Fraction** class below will contain two **int** attributes for the numerator and denominator of a fraction. The class will also contain a method **fractionToDecimal** that can be accessed from outside the class.

```
public class Fraction
{
    /* missing code */
    // constructor and other methods not shown
}
```

Which of the following replacements for `/* missing code */` is the most appropriate implementation of the class?

A.
```
public int numerator;
public int denominator;
private double fractionToDecimal()
{
    return (double) numerator / denominator;
}
```

B.
```
public double fractionToDecimal()
{
    private int numerator;
    private int denominator;
    return (double) numerator / denominator;
}
```

C.
```
public double fractionToDecimal()
{
    int numerator;
    int denominator;
    return (double) numerator / denominator;
}
```

D.
```
private int numerator;
private int denominator;
private double fractionToDecimal()
{
    return (double) numerator / denominator;
}
```

E.
```
private int numerator;
private int denominator;
public double fractionToDecimal()
{
    return (double) numerator / denominator;
}
```

14

Consider the following definition of the class Student.

```
public class Student
{
    private int grade_level;
    private String name;
    private double GPA;

    public Student (int g, String n, double gpa)
    {
        grade_level = g;
        name = n;
        GPA = gpa;
    }
}
```

Which of the following object initializations will compile without error?

A. **`Student chen = new Student (10, "Chen");`**
B. **`Student chen = new Student (3.75, "Chen", 10);`**
C. **`Student chen = new Student (10.0, "Chen", 3);`**
D. **`Student chen = new Student (10, "Chen", 3.75);`**
E. **`Student chen = new Student (10, 3.75, "Chen");`**

15

Consider the following class definition. Each object of the class Employee will store the employee's name as name, the number of weekly hours worked as **wk_hours**, and hourly rate of pay as **pay_rate**.

```
public class Employee
{
    private String name;
    private int wk_hours;
    private double pay_rate;

    public Employee(String nm, int hrs, double rt)
    {
        name = nm;
        wk_hours = hrs;
        pay_rate = rt;
    }

    public Employee(String nm, double rt)
    {
        name = nm;
        wk_hours = 20;
        pay_rate = rt;
    }

    public Employee(String nm)
    {
        name = nm;
        wk_hours = 20;
        pay_rate = 15.0;
    }

}
```

Which of the following code segments, found in a class other than **Employee**, could be used to correctly create an **Employee** object representing an employee who worked for 20 hours at a rate of $18.50 per hour?

```
I. Employee e1 = new Employee("Lili", 20, 18.5);
II. Employee e2 = new Employee("Steve", 18.5);
III. Employee e3 = new Employee("Carol");
```

A. **I only**
B. **III only**
C. **I and II only**
D. **I and III only**
E. **I, II, and III**

16

Consider the following method **substringFound**, which is intended to return true if a substring, key, is located at a specific index of the string phrase. Otherwise, it should return false.

```
public boolean substringFound(String phrase, String key, int index)
{
    String part = phrase.substring(index, index + key.length());
    return part.equals(key);
}
```

Which of the following is the best precondition for index so that the method will return the appropriate result in all cases and a runtime error can be avoided?

A. **`-1 < index < phrase.length()`**
B. **`-1 < index < key.length()`**
C. **`-1 < index < phrase.length() - key.length()`**
D. **`0 <= index < phrase.length() + key.length()`**
E. **`0 <= index < phrase.length() - index`**

17

The Fibonacci numbers are a sequence of integers. The first two numbers are 1 and 1. Each subsequent number is equal to the sum of the previous two integers. For example, the first seven Fibonacci numbers are 1, 1, 2, 3, 5, 8, and 13. The following code segment is intended to fill the fibs array with the first ten Fibonacci numbers. The code segment does not work as intended.

```
int[] fibs = new int[10];
fibs[0] = 1;
fibs[1] = 1;
for (int j = 0; j < fibs.length; j++)
{
    fibs[j+2] = fibs[j+1] + fibs[j];
}
```

Which of the following best identifies why the code segment does not work as intended?

- **A. In the for loop header, the initial value of j should be 1.**
- **B. In the for loop header, the initial value of j should be 2.**
- **C. The for loop condition should be j < fibs.length - 2.**
- **D. The for loop condition should be j < fibs.length + 2.**
- **E. The for loop should increment j by 2 instead of by 1.**

18

Consider the following code segment.

```
ArrayList<Integer> myList = new ArrayList();
for (int i = 0; i < 5; i++)
{
    myList.add(i + 1);
}
for (int i = 0; i < 5; i++)
{
    if (i % 2 == 0)
    {
        System.out.print(myList.get(i) + " ");
    }
}
```

What output is produced as a result of executing the code segment?

- **A. 0 1 2 3 4**
- **B. 1 2 3 4 5**
- **C. 0 2 4**
- **D. 1 3 5**
- **E. 2 4 6**

19

Assume that the classes listed in the Java Quick Reference have been imported where appropriate. Unless otherwise noted in the question, assume that parameters in method calls are not **null** and that methods are called only when their preconditions are satisfied. In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.
The following class keeps a record of items purchased by a customer and is able to produce a string formatted as a receipt.

```
public class Receipt
{
    // Instance variables.
    private String customerName;
    private String dateCode;
    private ArrayList<String> itemsPurchased;

    /** Constructs a Receipt object and initialized
     * all instance variables.
     */
    public Receipt(String n, String d)
    { /* to be implemented in part (a) */ }

    /** Adds an item String to itemsPurchased.
     */
    public void addItem(String i)
    { /* to be implemented in part (b) */ }

    /** Sorts the items in itemsPurchased alphabetically.
     */
    private void sortItems()
    { /* implementation not shown */ }

    /** Returns a string that can be printed as a receipt for
     * a customer.
     */
    public String formatReceipt()
    { /* to be implemented in part (c) */ }
}
```

A. Write the constructor method for the Receipt class that accepts two String objects and initializes all instance variables.

B. Write an *addItem* method that accepts a String and adds it to *itemsPurchased*.

```
Example of a String that might be returned by formatReceipt:

John Doe
Keyboard
Monitor
Mouse
Pencil
```

C. **Write a *formatReciept* method that returns a String that contains the customer's name followed by the list of purchased items (sorted alphabetically) on separate lines.**

20

Assume that the classes listed in the Java Quick Reference have been imported where appropriate. Unless otherwise noted in the question, assume that parameters in method calls are not **null** and that methods are called only when their preconditions are satisfied.

```
public class WordSearch
{
    // The list of words.
    private ArrayList<String> wordFragments;

    /** Constructs a WordSearch object as described in part (a).
    */
    public WordSearch(String word)
    { /* to be implemented in part (a) */ }

    /** Returns true if String word is found
     * in wordFragments.
    */
    public boolean isSimilar(String word)
    { /* to be implemented in part (b) */ }
}
```

Write the constructor for the **WordSearch** class. The constructor initializes the ArrayList **wordFragments** and adds fragments of the String **word** to **wordFragments**.
The following example shows the contents of **wordFragments** after a WordSearch object has been constructed with the String "hello".

Example:

```
WordSearch ws = new WordSearch( "java" );
```

After the **ws** object has been constructed, the **wordFragments** ArrayList in the **ws**WordSearch object will contain:

```
"j", "ja", "jav", "java”
```

A. Write the *WordSearch* constructor below.

```
public WordSearch(String word)
```

Write the method **isSimilar**. This method accepts a String **word** and compares it to each String in **wordFragments**. This method returns **true** if **word** is identical to any of the String fragments in **wordFragments**.

Example:

```
WordSearch ws = new WordSearch( "java" );
boolean match = ws.isSimilar( "ja" );
```

After this code has executed, **match** will contain the value `true`.

B. Write the *isSimilar* method below.

```
public boolean isSimilar(String word)
```

Optional Extra Credit (1 Point)

21

Write a LineSearch class as shown below. A LineSearch object contains the method `similarityScore(String[] words)` that returns the number of words that are similar to words in a line of text.

A new WordSearch object should be created for each word in the line of text and added to the ArrayList **words**.

When a String[] array is passed to the **similarityScore** method, each String in the String[] array should be passed to the **isSimilar** method of each WordSearch object in **words**. The score value returned by the **similarityScore** method is the sum of the number of words that are similar to words in the line of text (stored as WordSearch objects). Add **1** additional point for each pair of two words that are in the correct order.

Example:

```
LineSearch ls = new LineSearch( "This is a line of text" );

String[] searchWords = new String[] { "Th", "is", "test" };
int score = ls.similarityScore( searchWords );
```

After this code has executed, the variable **score** will contain the value **3** (2 similar words + 1 pair in correct order).

A. **Write the *LineSearch* class below.**

```
public class LineSearch
{
    private ArrayList<WordSearch> words;

    public LineSearch (String lineOfText)
    {













    }

    public int similarityScore(String[] searchTerms)
    {













    }
}
```