CSYE 6200 Exam – Sp 2019

1) Complete the following sentences: [6]

a) The three main principles of object-oriented programming are ___Inheritance___, ___Encapsulation____, and ______Polymorphism_____.

b) ___javac_____ compiles Java files into _____ Bytecode ____.

d) Java programs start in a method called ____main___.

2) Fill in the table: [6]

Primitive	Bits
float	32
double	64
byte	8
char	16
short	16
int	32
long	64

3) Write a for loop that counts from 10 to 40 stepping by 4: Answer(For):

```
for (int i = 10; i < 40 (i <= 40); i = i + 2 (i += 2)) { }
```

(+2 initialization, +2 comparison, +2 iteration)

4) Complete the following sentences. [4]

The ___extends____ keyword is used when inheriting from a regular class.

The ____implements___ keyword is used when inheriting from an interface.

5) After the following statements: [2]

```
double turnLength = 4.0;
float ropeLength = 20.5f;
int pCnt = (int) (ropeLength / turnLength);
System.out.println("pCnt = " + pCnt);
```

What will print on the console? ___ pCnt = 5___

6) After the showCount() method is executed: [2]

```
public class CTest {
    private int count = 8;
    public void showCount(int count) {
        if (true) count = 5;
        System.out.println("count = " + this.count);
    }}
```

What will print on the console if the method showCount() is called?
___count = 8____

7) Based on the code snippet below, what will display on the console?   [5]

```
int choice = 5;
int val = 0;
choice++;
switch (choice) {
    case 5: val = 50; break;
    case 6: val = 60;
    case 7: val = 70; break;
    default:
    case 8: val = 80; break;
}
System.out.println("val = " + val);
```

Answer: _____val = 70 (70[3])_______

8) Overloading methods is an example of which Object-Oriented Programming tenant?   [2]
Best answer: ____polymorphism_______

9) Overriding methods is an example of which Object-Oriented Programming tenant?   [2]
Best answer: _____inheritance_____

10) Complete the following sentence: [2]
A class with one or more abstract methods is said to be an _____abstract______ class.

11) After the following statements: [5]

```
int counter = 0;
boolean done = false;
while (!done) {
    if (counter == 2) continue;
    char outC = (counter < 3) ? ‘X’ : ‘O’;
    System.out.print(outC + “-“);
    counter++;
    if (counter == 4) break;
    if (counter > 4) done = true;
}
```

What will print on the console? ______X-X- (X-X [3])__________

12) For the following method, what should the last line be? [2]
(Note: write your answer in the space shown below.)

```
public double findArea(double width, double length) {
    double area = length * width;
    if (area < 0.0) area *= -1.0;
    ______return area____ ;

}
```

13) Complete the following sentences: [4]
The _____this_____ keyword is an implicit argument that is passed to each method call, providing a reference to the calling object.
Java defines one special superclass called ___Object______ that is the parent of all other classes.

14) Overloading constructors allows your code to hide internal working details. This technique of hiding internal object complexity is an example of which Object-Oriented Programming tenant? [2]
Answer: ___encapsulation______

15) Write a public method to calculate the total area. Finish the method shown below: [10]

```
public class DNAList {
    private String fragmentString = “ACGTGACAGT”;
    public void printReverse(String input) {
        int stringLen = input.length();
        if(stringLen == 0) return;
        System.out.print(input.charAt(stringLen - 1));
        printReverse(input.subString(0, stringLen - 1));
    }
}
```

16) Complete the following sentences [4]

In Java, the name of a method plus its parameter list is call a _____signature____ .

In Java, to prevent a method from being overridden, use the keyword ___final/static/private____.

17) For the following code, fill in the blanks so that the ProductRegistry class exhibits a Singleton design pattern: [10]

```
public class ProductRegistry {
    private String dataFileName = “dataFile.txt”;
    __ private ____ __ static__ ProductRegistry instance = null ;
    ___ private ____ ProductRegistry() {
    }
    public ____ static ____ ProductRegistry instance() {
    if (instance == null )
        instance = new ProductRegistry();
    return __ instance; __
    }
    public File getDataFile() throws IOException {
        File file = new File(dataFileName);
        return file;
    }
}
```

18) To create an instance of the ProductRegistry class (from Q. 17), and open the data file, what code should you write (fill in blanks below)? [6]

```
public class Z { …
       private ProductRegistry registry =
              __ProductRegistry.instance();_____
       public BufferedReader openDataFileReader() {
           try {
               File file = registry.getDataFile();
               return new BufferedReader(file);
           } catch (_____ Exception _____ ex) { … }
      } … (any reasonable exception is allowed)
```

19) Fill in the blanks [10]

a) Method overloading is one of the ways that Java implements the OO tenant of ___polymorphism/ encapsulation _____ .

b) When passing a on object instance to a method, Java uses pass-by-___Reference ________.

c) When a method calls itself, this is an example of ____recursion _______ .

d) The ___protected _______ keyword is used to make a member variable accessible by inheriting classes.

e) A static block is called when a class is ___loaded ______.

20) Use the following code, draw the UML class association diagram. [10]

```
public abstract class DriverRegistry extends Registry {
    protected String regID;
    private ArrayList<Driver> list = new ArrayList<Driver>();
    public void add(Driver driver) { list.add(driver); }
    public Driver getDriver(index) {
        int return (list.get(index));
    }
    public abstract void setReqId( String id );
… // rest of class omitted
}

public class CustomRegistry extends DriverRegistry implements
RegistryIoI {
    private VehicleRegistry vReg = VehicleRegistry.instance();
    @Override
    public Vehicle getVehicle( int index ) {
        Driver driver = getDriver(index);
        return (vReg.getVehicle(driver.getAltDriver());
    }
… // rest of class omitted
}
```

Answer - Draw the UML Class association diagram below:
(Note: interior class details may be omitted. Only draw associations between classes).

[+1 DriverRegistry]
[+1 CustomRegistry]
[+1 RegistryIoI]
[+2 inheritance association arrows]
[+1 Abstract/Interface notation]
[+1 Vehicle class w/weak association]
[+1 Driver class w/weak association]
[+2 Collection association for Drivers]