CS 0401 Fall 2019 Quiz 2 Version B
SOLUTIONS

1) Fill in the Blanks (14 points – 2 points each). Provide the MOST appropriate answers. If choices are given in square brackets, select one of the choices.

   a) The _____ SelectionSort ___________ algorithm finds the smallest item in an array and puts it into index 0, then finds the second smallest and puts it into index 1, etc.

   b) The _____ physical ___________ size of an ArrayList is the length of the underlying array and the _____ logical ___________ size of an ArrayList is the number of items being stored in the array.

   c) Composition gives a _____ has a ______ [is a, has a] relationship between the new and old classes while inheritance gives a _____ is a ______ [is a, has a] relationship between the new and old classes.

   d) The Java classes Integer, Float, etc. are called _____ Wrapper ___________ classes and are used to allow the primitive types to be accessed as objects.

   e) Instance data in Java declared to be _____ protected ___________ can be directly accessed within its own class, any subclasses of that class and the package in which that class is located.

   f) A Java _____ interface ___________ is more or less a set of method headers with no method bodies and no instance variables.

   g) Ad hoc polymorphism utilizes method _____ overloading ___________ [overloading, overriding] whereas subclassing polymorphism utilizes method _____ overriding ___________ [overloading, overriding]

2) True / False (12 points – 2 points each). Explain false statements for full credit.

   a) A separate copy of a Java static variable is created for each object of a given class.  
      False – only one copy of the variable exists and it is shared.

   b) To "increase the size" of a full array in Java we must actually make a new, larger array and copy the data from the old array to the new one.  
      True

   c) The binary search algorithm of an array requires the data in the array to be sorted.  
      True

   d) In a Java Person class, the age of the Person should be an instance variable since it is a fundamental attribute of the Person.  
      False – age can change over time without being explicitly mutated.

   e) Consider classes Foo and SubFoo, where SubFoo is a subclass of Foo.  The Java statement:
      
      Foo F = new SubFoo();  
      is legal  
      True

   f) Consider classes Foo and SubFoo, where SubFoo is a subclass of Foo.  The Java statement:
      
      SubFoo S = new Foo();  
      is legal  
      False – isa only works from subclass to superclass.
3) **Trace (14 points).** Consider the SBHolder class on this page and the Trace1B program on the next page. Give all output, in the **correct order** and format, produced by execution of the Trace1B Java program. Use the bottom of the next page for your output. Clearly indicate your output by drawing a box around it. Be careful! I recommend drawing pictures of your objects and references.

```java
public class SBHolder
{
    private StringBuilder sb1;
    private StringBuilder sb2;

    public SBHolder(String s1, String s2)
    {
        sb1 = new StringBuilder(s1);
        sb2 = new StringBuilder(s2);
    }

    public SBHolder(SBHolder old)
    {
        sb1 = old.sb1;
        sb2 = new StringBuilder(old.sb2);
    }

    public String toString()
    {
        return sb1.toString() + "", " + sb2.toString();
    }

    public void mutate(String s1, String s2)
    {
        sb1.append(s1);
        sb2.append(s2);
    }

    public StringBuilder getsb1()
    {
        return new StringBuilder(sb1);
    }

    public StringBuilder getsb2()
    {
        return sb2;
    }
}
```
public class Trace1B
{
    public static void main(String[] args)
    {
        SBHolder H1 = new SBHolder("First", "One");
        System.out.println(H1.toString());
        StringBuilder data1 = H1.getsb1();
        StringBuilder data2 = H1.getsb2();
        data1.append("Second");
        data2.append("Two");
        H1.mutate("Third", "Three");
        System.out.println(H1.toString());
        System.out.println(data1.toString() + ", " + data2.toString());
        System.out.println();
        H1 = new SBHolder("Left", "Right");
        System.out.println(H1.toString());
        SBHolder H2 = new SBHolder(H1);
        H1.mutate("Side", "Hand");
        H2.mutate("Wing", "Field");
        System.out.println(H1.toString());
        System.out.println(H2.toString());
    }
}

Answer:
First, One
FirstThird, OneTwoThree
FirstSecond, OneTwoThree

Left, Right
LeftSideWing, RightHand
LeftSideWing, RightField

Notes:
getsb1() returns a copy of the StringBuilder in the SBHolder object, while getsb2() returns a reference to the actual object. Thus, data2 is a StringBuilder within the H1 object while data1 is not.

The copy constructor makes a copy of sb2 but does not copy sb1. Thus the sb1 references in H1 and H2 are accessing the same object.
public class SuperClass
{
    protected StringBuilder A;

    public SuperClass(String x)
    {
        A = new StringBuilder(x);
    }

    public String toString()
    {
        return "Data is: " + A.toString();
    }

    public void mutate(String data)
    {
        A.insert(0, data);
    }
}

public class SubClass extends SuperClass
{
    public SubClass(String x)
    {
        super(x);
    }

    public String toString()
    {
        return "Subclass version: " + super.toString();
    }

    public void mutate(String data)
    {
        A.append(data);
    }
}
public class Trace2B {
    public static void main(String [] args) {
        SuperClass [] Arr = new SuperClass[3];
        Arr[0] = new SuperClass("Inigo");
        Arr[1] = new SubClass("Miracle");

        for (SuperClass S: Arr) {
            System.out.println(S.toString());
            System.out.println();
            Arr[0].mutate("Montoya");
            Arr[1].mutate("Max");
            Arr[2].mutate("Humperdinck");
            for (SuperClass S: Arr) {
                System.out.println(S.toString());
            }
        }
    }
}

Answer:
Data is: Inigo
Subclass version: Data is: Miracle
Data is: Prince
Data is: MontoyaInigo
Subclass version: Data is: MiracleMax
Data is: HumperdinckPrince

Notes:
The toString() method in SubClass is overridden to add "Subclass version:" at the beginning of its string value. The mutate() method in SubClass is overridden to append at the end of the StringBuilder rather than insert at the front. Through polymorphism, the different versions of the methods are called when the Arr array is accessed.