

1. Matching [20 pts]

Choose the **best** single definition for each term.

1. _____ protected
 2. _____ super
 3. _____ overriding
 4. _____ overloading
 5. _____ abstract class
 6. _____ interface
 7. _____ java.lang.Object
 8. _____ JRadioButton
 9. _____ JCheckBox
 10. _____ perfect hash function
- A.** A swing component which allows a user to select one item from a group.
 - B.** A swing component which allows a user to select multiple items from a group.
 - C.** A reserved word which allows a class to refer to its parent class.
 - D.** A reserved word which allows a class to refer to the root of the class hierarchy.
 - E.** The base class from which all other classes are derived.
 - F.** The recommended base class that you should derive from.
 - G.** A collection of only abstract methods and constants.
 - H.** A class that cannot be instantiated, only extended.
 - I.** The condition where a subclass declares a method with same signature as in a superclass.
 - J.** The condition where a class declares a method with the same name, but different parameter list.
 - K.** A function applied to a class such that a unique code is produced for each different instance of the class.
 - L.** A function applied to a class such that a code is produced which fits a stochastic probability function distribution.
 - M.** A visibility modifier which allows access to instance data and methods only by subclasses.
 - N.** A visibility modifier which allows access to instance data and methods by subclasses and by other classes in the same package.

2. Constructor Tracing[12 pts]

What is printed by the following code.

```
public class Parent {
    protected String name;
    public Parent(String name) {
        this.name=name+" P";
        System.out.println("In Parent "+name);
    }
    public Parent(){
        this("Default");
        System.out.println("In Parent "+name);
    }
    public String toString(){
        return "Name is "+name;
    }
}

public class Child extends Parent {
    private String name;
    public Child(String name, String name2) {
        super(name);
        this.name=name2;
        System.out.println("In Child with "+name);
    }
    public Child(String name) {
        this("Default",name);
        System.out.println("In Child with "+this.name);
    }
    public String toString() {
        return "Names are: "+this.name+" "+super.name+" "+name;
    }
    public static void main(String[] args){
        Child c = new Child("Dora");
        Parent p = new Parent();
        Parent p1 = c;
        System.out.println(p);
        System.out.println(c);
        System.out.println(p1);
    }
}
```

Output:

3. Polymorphism [12 pts]

Consider the following code:

```
public abstract class Airplane {
    protected int runTime;
    public abstract void load();
    public abstract void fly();
}

public class Jet extends Airplane {
    public void load() { }
    public void fly() { }
    public void zoom() { }
}

public interface PropDriven{
    public void adjust();
}

public class Biplane extends Airplane implements PropDriven {
    public void load() { }
    public void fly(){ }
    public void adjust() { }
}

public class WWI extends Biplane {
    public void fly(){ }
    public void shoot(){ }
}
```

For each of the following lines of code, decide whether they will compile and run without error. If the line(s) are OK, then simply write OK. If they will fail to compile, write COMPILE ERROR or if they will fail at runtime write RUNTIME ERROR. For those items that are not OK, give a BRIEF explanation of what is wrong.

2 (a) PropDriven pd = new WWI();

2 (b) Airplane a = new Airplane();

- 2 (c) `Airplane a = new WWI();`
`a.fly();`
- 2 (d) `Airplane[] a= {new WWI(), new Biplane(), new Jet()};`
`a[2].zoom();`
- 2 (e) `WWI fighter = new Biplane();`
`fighter.shoot();`
- 2 (f) `Airplane a = new Biplane();`
`((WWI)a).shoot();`

4. **Dynamic Binding [11 pts]**

Assume you are given the following classes:

- Vehicle is abstract, and has the abstract method `public int cargo()`, a method which returns the cargo in pounds that a vehicle carries.
- SemiTruck extends Vehicle and implements cargo so that it returns 10000.
- Pickup extends Vehicle and implements cargo so that it returns 1200.
- Van extends Vehicle and implements cargo so that it returns 2000.

Write the method `public int totalCargoCapacity(Vehicle[] fleet)` which computes the total pounds of cargo capacity available for all the vehicles in a company fleet. This method should be useable with other subclasses of Vehicle that might be created later. For this method you may not use:

1. The `instanceof` operator
2. Casting of any sort
3. The `getClass()` method

If you violate the above restrictions, you will receive no credit for this question. For example, if a fleet were created with

```
Vehicle[] fleet = {new SemiTruck(), new Van(), new Pickup()};
```

Then the call: `totalCargoCapacity(fleet)` would return 13200.

```
public int totalCargoCapacity(Vehicle[] fleet) {
```

```
}
```

5. Exceptions and Arrays [15 pts]

Assume you have available the checked exception *NumTooSmallException* and the unchecked exception *ArrayOutOfOrderException*. Write a method *checkOrder* which takes in an array of int and checks whether the array is in ascending order (i.e. for each int at position *i*, it is less than or equal to the int at position *i+1*). The method returns nothing. If any element in the array is less than 0, your method should throw a *NumTooSmallException*. If the array is out of order, throw an *ArrayOutOfOrderException*. (Your method does not handle these exceptions, it simply throws them.)

6. File IO [15 pts]

- (a) Character streams manage
- A) byte-sized data
 - B) binary data
 - C) Unicode characters
 - D) ASCII characters
 - E) compressed data
- (b) To input from a text disk file, you should use the _____ class.
- A) InputStreamReader
 - B) FileReader
 - C) Keyboard
 - D) System.in
 - E) TextReader
- (c) The standard sequence of using a stream is:
- A) open, read or write (all streams auto-close)
 - B) read or write, close (all streams auto-open)
 - C) read or write (all streams auto-open and close)
 - D) open, read or write, close
- (d) Assume infile is a BufferedReader for a textfile and that the textfile is empty. What is returned from the method call `infile.readLine()`; ?
- A) 0
 - B) null
 - C) a special character known as the End-of-file marker (EOF)
 - D) none of the above, the message causes a `NullPointerException` to be thrown
 - E) none of the above, the message causes a `EndOfFileException` to be thrown
- (e) When programming using the Java streams, which of the following is NOT true:
- A) You must catch or have a throws clause in the method header for `IOExceptions` in any method that involves use of a stream class.
 - B) You must choose the proper stream class to use based upon the type data you are going to process (binary or text).
 - C) You must choose the proper stream class to use based upon whether you are doing input or output.
 - D) You must implement the `Serializable` interface for all classes that intend to perform any IO operations.

7. Hashtable Coding [15 pts]

You are writing a Hashtable class which uses linear probing for collision handling and stores strings. Write the printMe method, which prints the contents of the hashtable, in the following format:

```
Bucket 0 empty
Bucket 1 Bear
Bucket 2 Alice
Bucket 3 Junior
Bucket 4 empty
Table Capacity = 5
Occupied = 3
Load Factor = .6
```

Only the methods shown below are implemented. You may use any implemented methods in your method.

```
public class Hashtable {
    Bucket[] table;
    private class Bucket {
        public String data;
        public boolean deleted;
        public Bucket(String d) {data=d;deleted=false;}
    }
    public boolean isEmpty() { } //check if any entries
    private void grow(){ } //increase size if full
    public int size() { } //return number of entries
    public void add(String s) { } //add something to table
    public boolean remove(String s) { } //remove something false if fail
    public int capacity(){ } //return total capacity of hashtable
    //your method
    public void printMe(){/*fill in this method*/}
}
```

Hashtable Coding, Continued