Name: ________________________

GWU email id: __________________

CS2113 Exam 1

Spring 2023

Write your name on the top of this exam and do not open the packet until told to do so by the professor.

45 minutes.

_________ / 27 points

Assume all code compiles unless states otherwise.

1. Give two different things we discussed in class about how writing software was different in the 1950s versus today? [2pts]
   a. Reason 1:

   b. Reason 2:

2. Short answer (two or less phrases): [14 pts]
   a. What OOP concept, as discussed in class, does `private` visibility allow us to achieve?

   b. How is encapsulation different from information hiding?

   c. Draw a UML diagram where a `Tiger` class inherits from a `Cat` class.

   d. Why would you use an interface rather than an abstract class?

   e. Why would you use an abstract class rather than an interface (don't just "flip" your answer to d)?

   f. What are the two methods of the Iterator interface?
      i. Method 1:

      ii. Method 2:

   g. What is the point of having exception handling, as discussed in class?

3. Modify the class below (just insert or cross parts out) to make it work with generics (that is, to work with any kind of object, not just `int`s as below): [4 pts]

```
public class List           {
    public int get(int idx);
    public void set(int idx, int val);
    public void insert(int idx, int val);
    public void addToHead(int val);
    public void addToTail(int val);
    public boolean empty();
    public int size();
}
```

4. Now complete the code below to create a `List` object that only stores `Person` objects, and adds a single person to the `List`, and then pulls it out of the `List`. You may assume all classes have default constructors: [4 pts]

```
_____________________ employees = new ____________________();

employees.addToHead(_________________________________);

Person first = ________________________________;
```

(more on back)

5. Consider the three compiling files below [3pts]:

```
/*Pair.java*/
public class Pair{
  private int left,right;
  public Pair(int l,int r){left=l;right=r;}
  public int getLeft(){return left;}
  public int getRight(){return right;}
  public String toString(){return "("+left+","+right+")";}
}
```

```
/*LabPair.java*/
public class LabPair extends Pair{
  private String label;
  public LabPair(int l, int r, String lab){
     super(l,r);
     label=lab;
  }
  public String getLabel(){return label;}
  public String toString(){return label+":"+super.toString();}
}
```

```
/*Main.java*/
public class Main{
  public static void main(String args[]){
     Pair p[3] = {new Pair(1,2),
                  new LabPair(3,4,"A"),
                  new LabPair(5,6,"B")};
    for(int i=0;i<3;i++){
      System.out.println(p[i]);
    }
  }
```

a. Circle an example of polymorphism/dynamic binding and explain it below:

SCRATCH PAPER