CS 3443-X
Application Programming

SAMPLE Midterm Exam
Semester 20XX

Name: _____

UTSA ID: _____
abc123

This is a closed book, closed notes exam. Collaboration, materials, and all electronics are strictly prohibited. If more space is needed to answer a question, use the back of a page and indicate the question number with your answer.

Have a question? Raise your hand and the proctor will come to you.

I. (5 pts.) What does the following code fragment print?

```
int[] array = {1,2,3,2,1};
int v = 1;

for( int val : array ){
  if( v > val )
        v += val;
  else
        v -= val;

  System.out.printf("%s", val );
}

System.out.println("");
System.out.printf("%d", v );
```

II. (5 pts.) What does the following code fragment print?

```
char[] letters = {'A', 'B', '2', 'L', 'U', 'H', 'K', 'm', 'G', 'c', 'Z'};
int[]  numbers = {3, 4, 9, 6};
String printMe = "";

for( int index : numbers ){
     printMe += letters[index];
}

System.out.println( printMe + "!" );
```

III. (25 pts.) Consider the main method in the `LibraryTest` class below for testing the `Book` class on this page and the `Library` class on the next page. Complete the `Library` class so that the main method of `LibraryTest` will print out:

```
The UTSA Library collection includes:
  (1) "Effective Java" by Joshua Bloch
  (2) "Sherlock Holmes" by Arthur Conan Doyle
"Sherlock Holmes" is available.
"Sherlock Holmes" is not available.
```

All methods called in the main method must be implemented.

```
public class LibraryTest{

     public static void main( String[] args ){

          Library lib = new Library( "UTSA Library" );

          Book book1 = new Book("Effective Java", "Joshua Bloch");
          Book book2 = new Book("Sherlock Holmes", "Arthur Conan Doyle");
          lib.addBook( book1 );
          lib.addBook( book2 );

          Book book3 = new Book("Sherlock Holmes", "Arthur Conan Doyle");
          boolean onShelf = lib.isAvailable( book3 );

          System.out.println( lib );
          System.out.println( "\"" + book3.getName() + "\""
                         + (onShelf ? "is" : "is not") + " available" );

          lib.checkOut( book3 );
          onShelf = lib.isAvailable( book3 );
          System.out.println( "\"" + book3.getName() + "\""
                         + (onShelf ? "is" : "is not") + " available" );
     }

}
```

```
public class Book {
     private String name;
     private String author;

     public Book( String n, String a ){
          this.name = n;
          this.author = a;
     }

     public String getName(){
          return this.name;
     }
}
```

```
import java.util.*;

public class Library {

     private String name;
     private ArrayList<Book> books;









}
```

IV. (20 pts.) The following main method uses the `Generator` interface and the `AbstractGenerator` and `LetterGenerator` classes on the following pages. What does it print out?

```
public class GeneratorTest {
     public static void main( String[] args ){

        Generator gen = new LetterGenerator("one","twenty",2);

        String t=" 3”;

        for( int i=0; i<gen.length(); i++ ){
              if( i==1 || i==2 || i==5 || i==7 )
                    t = " 4";
              else t = " 3";
              System.out.println( gen.next() + t );
        }
     }
}
```

```
/**
 * A Generator supplies a series of values from a known list.
 * @author Amanda Danko
 */
public interface Generator {
     /**
      * @return the next value generated
      */
     public char next();


     /**
      * @return the length
      */
     public int length();

}
```

```
/**
 * An AbstractGenerator gets the next value & returns value at an index.
 * @author Amanda Danko
 */
public abstract class AbstractGenerator implements Generator{
     private int index;

     public abstract char getValueAt(int index);

     public char next(){
        index++;
        return getValueAt(index-1);
     }
}
```

```
/**
 * A LetterGenerator generates letters.
 * @author Amanda Danko
 */
public class LetterGenerator extends AbstractGenerator {
     private String a,b,c;

     public LetterGenerator( String x, String y, int z ){
        this.a = x;
        this.b = y;
        this.c = String.valueOf( z );
     }

     public char getValueAt(int index){
        return this.a.charAt( index );
     }

     public int length(){
        return a.length();
     }
}
```

V. (25 pts.) Suppose the first line in the GeneratorTest main method is changed to:

```
Generator gen = new NumberGenerator(4457.2, 135.7, 3043.2);
```

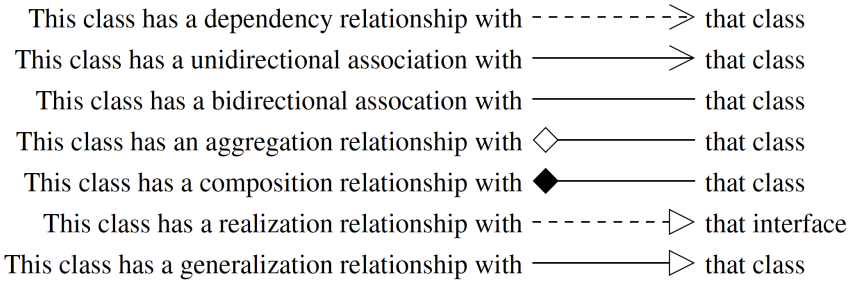
Implement the `NumberGenerator` class so that the main method prints out:

```
4 3
4 4
5 4
7 3
. 3
2 4
```

Hint: you may either extend AbstractGenerator or directly implement Generator.

VI. (10 pts.) Draw a UML class diagram that shows the relationship among `Library`, `LibraryTest`, and `Book`.

The different class relationships are shown below:



VII. (10 pts.) Draw a UML class diagram that shows the relationship among `Generator`, `AbstractGenerator`, `NumberGenerator`, and `LetterGenerator`.