THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2010
Campus: City

COMPUTER SCIENCE

TEST

Principles of Programming

(Time Allowed: 75 minutes)

Note:

- The use of calculators is NOT permitted.
- Compare the test version number on the Teleform sheet supplied with the version number above. If they do not match, ask the supervisor for a new sheet.
- Enter your name and student ID on the Teleform sheet. Your name should be entered left aligned. If your name is longer than the number of boxes provided, truncate it.
- Answer Section A (multiple-choice questions) on the Teleform answer sheet provided. Answer Section B in the space provided in this booklet. Attempt all questions.
- For Section A, use a dark pencil to mark your answers in the answer boxes on the Teleform sheet. Check that the question number on the sheet corresponds to the question number in this question/answer book. Do not cross out answers on the Teleform sheet if you change your mind. You must completely erase one answer before you choose another one. If you spoil your sheet, ask the supervisor for a replacement. There is one correct answer per question.
- For Section B, write your answers in the spaces provided in this booklet. Write as clearly as possible. The space provided will generally be sufficient but is not necessarily an indication of the expected length. Extra space is provided at the end of this exam book.
- An appendix with a simplified API is included on the last page.

Surname:	
First Name(s):	
Student ID:	
Login Name (UPI):	

Your marked test script will be returned during the weekly lab sessions. In the box below, clearly write the **DAY OF THE WEEK** and the **STARTING TIME** of the lab session where you would like your test script returned.

LAB TIME: *where your marked test will be returned*	*Day of week* *Starting time of lab*

MARKERS ONLY

Question	Marks	Out Of
1 - 20		40
21		13
22		16
23		18
24		13
Total		**100**

1) What is the output of the following code?

```
int a = 5;
int b = 10;
int c;

c = a;
a = b;
b = c;

System.out.println(a + " " + b + " " + c);
```

(a) `10 5 10`

(b) `10 5 5`

(c) `5 10 10`

(d) `5 10 5`

(e) `10 10 5`

2) What output is produced when the following code is executed?

```
int x = 9;
int y = 4;
String s = " + ";

System.out.println(x - y + s + y + y);
```

(a) `5 + 8`

(b) `5 + 44`

(c) `13`

(d) `49`

(e) None of the above

3) What output is produced when the following code is executed?

```
int a, b;
double c, d;

a = 5;
b = a + 2;
c = b + 2;
d = c + 2;

c = c / 2 * 2;
b = b / 2 * 2;

System.out.println("a: " + a + " b: " + b + " c: " + c + " d: " + d);
```

(a) `a: 5 b: 7 c: 9.0 d: 11.0`

(b) `a: 5 b: 6 c: 9.0 d: 11`

(c) `a: 5 b: 6 c: 9 d: 11.0`

(d) `a: 5 b: 7 c: 9.0 d: 11`

(e) None of the above

4) What output is produced when the following code is executed?

```
String phrase = "ORIGIN OF LIFE";

String part1 = phrase.substring(3, 6);
String part2 = phrase.substring(11);

String result = "" + part1 + part2;

System.out.println(result);
```

(a) `GINIFE`

(b) `GIN IFE`

(c) `IGINFE`

(d) `IGINLIFE`

(e) None of the above

5) What output is produced when the following code is executed?

```
String phrase = "ORIGIN OF LIFE";

char c1 = phrase.charAt(phrase.length() - 1);
char c2 = phrase.charAt(1);

String result = "" + c1 + c2;

System.out.println(result);
```

(a) `FR`

(b) `ER`

(c) `FO`

(d) `EO`

(e) None of the above

6) What output is produced when the following code is executed?

```
String phrase = "ORIGIN OF LIFE";

int position1 = phrase.indexOf("I");
int position2 = phrase.indexOf("Z");

int sum = position1 + position2;

System.out.println(sum);
```

(a) `3`

(b) `10`

(c) `1`

(d) `2`

(e) None of the above

7) Consider the following segment of code:

```
boolean a = 5 < 6;      // statement 1
int b = 5 || 6;         // statement 2
```

Given the type definitions of the variables a and b, which of the following statements is true:

(a) statement 1 will not compile, but statement 2 will compile correctly

(b) both statements will compile, but there will be an error if the code is executed

(c) neither statement will compile correctly

(d) both statements will compile correctly, and the code will execute without error

(e) statement 1 will compile, but statement 2 will not compile correctly

8) Given the following method definition:

```
private boolean myMethod(int a, int b, String c) {
      int d = 20;
      int e = a + b;

      if (e % 2 == 0 && c.length() == 2) {
            return true;
      }

      return false;
}
```

which of the following method calls will compile?

(a) `int result = myMethod(5, 2, "HI");`

(b) `boolean result = myMethod(5, "HI", 2);`

(c) `boolean result = myMethod(5, 2, "HI");`

(d) `int result = myMethod(5, 2, HI);`

(e) None of the above

9) Which of the following outputs could not possibly have been produced by the following code?

```
String word = "TINKLE";
String word2 = word;
int position;

while (word2.equals(word)) {
      position = (int)(Math.random() * word2.length());
      word2 = word2.substring(position) + word2.substring(0,
                                                          position);
}

System.out.println(word2);
```

(a) `KLETIN`

(b) `TINKLE`

(c) `ETINKL`

(d) `NKLETI`

(e) None of the above

10) Which of the following loops **DOES NOT** run 10 times?

Note: Only the relevant parts of the loop implementation are shown

(a)
```
int i=1;
while (i<10) {
      i = i + 1;
      ...
```

(b)
```
for (int i=0; i<10; i++) {
      ...
```

(c)
```
int i=0; for (; i<10; i++) {
      ...
```

(d)
```
int i=0;
while (i<=9) {
      i = i + 1;
      ...
```

(e)
```
int i; for (i=0; i<=9; i++) {
      ...
```

11) Which of the following would **NOT** be acceptable as the condition in a `for`-loop?

(a) `(x != 10) && (y > 5)`

(b) `x < 10`

(c) `x > y`

(d) `x == 10`

(e) `x = x + 1`

12) Consider the following code:

```
int i = 0;

for (i = 100; i >= 0; i--) {
    System.out.println(i);
}

System.out.println(i);  // Line X
```

What is printed by the code in **Line X**?

(a) The string "`i`"

(b) The value of the variable `i` before the start of the loop

(c) 0

(d) Nothing, because i is out of scope

(e) -1

13) Consider the following string declaration and initialization:

```
String s = "long white cloud";
```

Which of the following expressions evaluates to the String "`long cloud`"?

(a) `s.substring(0,5) + s.substring(11)`

(b) `s.substring(0,4) + "cloud"`

(c) `s.substring(0,4) + s.substring(10,15)`

(d) `s.indexOf("long") + s.indexOf("cloud")`

(e) `s.substring("long") + " " + s.substring("cloud")`

14) Which of the following methods would you use to extend an array of integers by one element?

(a)
```
private int[] extendArray(int[] shortArray, int newElement) {
      int[] newArray = new int[shortArray.length + 1];
      for (int i = 0; i<shortArray.length; i = i + 1) {
            newArray[i] = shortArray[i];
      }
      newArray[shortArray.length] = newElement;
      return newArray;
}
```

(b)
```
private int extendArray(int[] shortArray, int newElement) {
      int[] newArray = new int[shortArray.length + 1];
      newArray = shortArray;
      newArray[shortArray.length + 1] = newElement;
      return newArray;
}
```

(c)
```
private int[] extendArray(int[] shortArray, int newElement) {
      int[] newArray = new int[shortArray.length];
      for (int i = 0; i<=shortArray.length; i = i + 1) {
            newArray[i] = shortArray[i];
      }
      newArray[shortArray.length] = newElement;
      return newArray;
}
```

(d)
```
private int[] extendArray(int[] shortArray, int newElement) {
      int[] newArray = new int[shortArray.length + 1];
      for (int i = 0; i<=shortArray.length; i = i + 1) {
            newArray[i] = shortArray[i];
      }
      newArray[shortArray.length + 1] = newElement;
      return newArray;
}
```

(e)
```
private int[] extendArray(int[] shortArray, int newElement) {
      int[] newArray = new int[shortArray.length];
      for (int i = 0; i<shortArray.length; i = i + 1) {
            newArray[i] = shortArray[i];
      }
      newArray[shortArray.length] = newElement;
      return newArray;
}
```

15) To check whether two arrays contain the same values in the same order, you would

(a) compare the two array variables using the equals() method

(b) convert the arrays into strings and compare the strings with the equals() method

(c) compare the two arrays element by element in a for-loop

(d) compare the two array variables using the != operator

(e) compare the two array variables using the == operator

16) What is the output of the following code?

```
int i = 10;

while (i < 20) {
      if (i % 2 == 1) {
            System.out.print("X");
      } else {
            System.out.print("O");
      }
      i = i + 2;
}
```

(a) XOXOXOXOXO

(b) XXXXX

(c) XOXOX

(d) OOOOO

(e) OXOXOXOXOX

17) Consider the following code (notice that the values assigned to variables a and b have been removed):

```
boolean a = ???;
boolean b = ???;
boolean result = (!a && b) && b;
```

What values should be assigned to variables a and b such that the variable result will store the value true?

(a)
```
boolean a = true;
boolean b = true;
```

(b)
```
boolean a = false;
boolean b = false;
```

(c)
```
boolean a = true;
boolean b = false;
```

(d)
```
boolean a = false;
boolean b = true;
```

(e) None of the above

18) What output is produced by the following code?

```
int x = 10;
int y = 5;

if (x < 10) {
      if (y != 5) {
            System.out.println("aaa");
      } else {
            System.out.println("bbb");
      }
} else if (y > 10) {
            System.out.println("ccc");
} else {
      if (y != 5) {
            System.out.println("ddd");
      } else {
            System.out.println("eee");
      }
}
```

(a) `ddd`

(b) `eee`

(c) `bbb`

(d) `aaa`

(e) `ccc`

19) What is the output when the start() method is executed?

```
public void start() {
      int result;

      result = conditionals(0, 0);
      System.out.println(result);

      result = conditionals(-2, -5);
      System.out.println(result);

      result = conditionals(4, -1);
      System.out.println(result);
}

private int conditionals(int a, int b) {
      if( a > 0 && b <= 0 ) {
            return 100;
      } else if( a > 0 || b > 0 ) {
            return 200;
      } else {
            return 300;
      }
}
```

(a)
```
100
200
300
```

(b)
```
300
200
100
```

(c)
```
100
300
200
```

(d)
```
200
100
300
```

(e)
```
300
300
100
```

20) Consider the following code which includes a conditional statement. The values assigned to the variables are missing from the code and have been replaced with **????:**

```
int a, b, c;

a = ????;
b = ????;
c = ????;

if (!(a > b || b != c)) {
      System.out.println("success");
}
```

Which of the following set of value assignments could be used to replace the **????** above, such that "`success`" would be printed out?

(a)
```
a = 10;
b = 20;
c = 30;
```

(b)
```
a = 10;
b = 10;
c = 10;
```

(c)
```
a = 20;
b = 10;
c = 30;
```

(d)
```
a = 20;
b = 10;
c = 10;
```

(e)
```
a = 30;
b = 30;
c = 20;
```

SECTION B

Answer all questions in this section in the space provided. If you run out of space then please use the Overflow Sheet and indicate in the allotted space that you have used the Overflow Sheet. (An appendix with a simplified API is included on the last page. You may detach this appendix.)

Question 21 [13 marks]

a) Bug hunting time! The following method will not compile.

```
private int charactersLeftInStringAfter(String needle, String haystack) {
      // Compute the number of characters in haystack after
      // the end of the string given by the needle parameter:

      int needlePosition = haystack.indexOf(needle);

      if (needlePosition > -1) {
            // needle found in haystack
            return haystack.length() - needlePosition - needle.length();
      } else {
            System.out.println(needle + "not found in " + haystack);
      }
}
```

Describe why this code will not compile and clearly state what should be done to fix it.

(6 marks)

b) Complete the `addWordIfNotAlreadyInString()` method which is passed two String parameters. The first String parameter is a list of words separated by a space and the second parameter is a single word. The method returns a String containing all the words of the first parameter with the single word followed by a blank space added to the end of the String. This addition only happens if the word is not already present in the String.

Note: The parameter, `wordsSoFar`, always starts and ends with a blank space.

For example, the following code:

```
String words = " dog catfish hen pig ";

words = addWordIfNotAlreadyInString(words, "bird");
System.out.println("1. " + words);

words = addWordIfNotAlreadyInString(words, "cat");
System.out.println("2. " + words);

words = addWordIfNotAlreadyInString(words, "bee");
System.out.println("3. " + words);

words = addWordIfNotAlreadyInString(words, "pig");
System.out.println("4. " + words);
```

would print:

```
1.  dog catfish hen pig bird
2.  dog catfish hen pig bird cat
3.  dog catfish hen pig bird cat bee
4.  dog catfish hen pig bird cat bee
```

```
private String addWordIfNotAlreadyInString(String wordsSoFar, String word){

}
```

(7 marks)

Question 22 [16 marks]

Consider the following method named `maxOfThree()`. However, notice that the `return` statements are incomplete in this method definition.

```
private int maxOfThree(int a, int b, int c) {

      if (a < b) {
            if (b < c) {
                  return ??? 1 ???
            } else {
                  return ??? 2 ???
            }
      } else if (a < c) {
            return ??? 3 ???
      } else {
            return ??? 4 ???
      }
}
```

This method accepts three input values (`a`, `b` and `c`) and it should return the largest of the 3 values.

a) In the space provided below, indicate what the correct return statements should be so that this method would correctly return the largest of the input values.

```
private int maxOfThree(int a, int b, int c) {

      if (a < b) {
            if (b < c) {
                  return [          ] ;
            } else {
                  return [          ] ;
            }
      } else if (a < c) {
            return [          ] ;
      } else {
            return [          ] ;
      }
}
```

(6 marks)

b) The `maxOfThree()` method can be re-written without using any conditional statements (e.g. without using any `if`, `else if`, or `else` statements). In the space provided below, re-write the `maxOfThree()` method but **DO NOT** use any conditional statements.

Hint: The `Math.max()` method will be useful.

```
private int maxOfThree(int a, int b, int c) {




}
```

(5 marks)

c) Consider the following two method definitions:

```
private void changeX(int x) {
      x = x + 1;
}

private int changeY(int y) {
      y = y + 1;
      return y;
}
```

What would be produced as output when the following start() method is executed?

```
public void start() {

      int a, b, x, y;

      a = 1;
      b = 2;
      x = 3;
      y = 4;

      changeX(a);
      changeX(x);

      System.out.println("a: " + a);
      System.out.println("x: " + x);

      b = changeY(b);
      y = changeY(y);

      System.out.println("b: " + b);
      System.out.println("y: " + y);

}
```

```
a:

x:

b:

y:
```

(5 marks)

Question 23 [18 marks]

a) Complete the `getRandomLetter()` method which returns a String. The method returns either the String "A", "B", "C" or "D". Each letter has an equal chance of being returned by the method. You will need to use `Math.random()` in this question.

```
private String getRandomLetter() {

        String letter1 = "A";
        String letter2 = "B";
        String letter3 = "C";
        String letter4 = "D";

}
```

(6 marks)

b) Complete the `printWithBlanks()` method which is passed a String parameter, `word`. The method prints the `word` with a space between each character. For example, the following code:

```
printWithBlanks("WHOOSH");
printWithBlanks("RATATTAT");
printWithBlanks("ELEPHANT");
```

would print:

```
W H O O S H
R A T A T T A T
E L E P H A N T
```

Note: It is OK if a blank space is printed at the end of the word.

```
private void printWithBlanks(String word){

        System.out.println();
}
```

(6 marks)

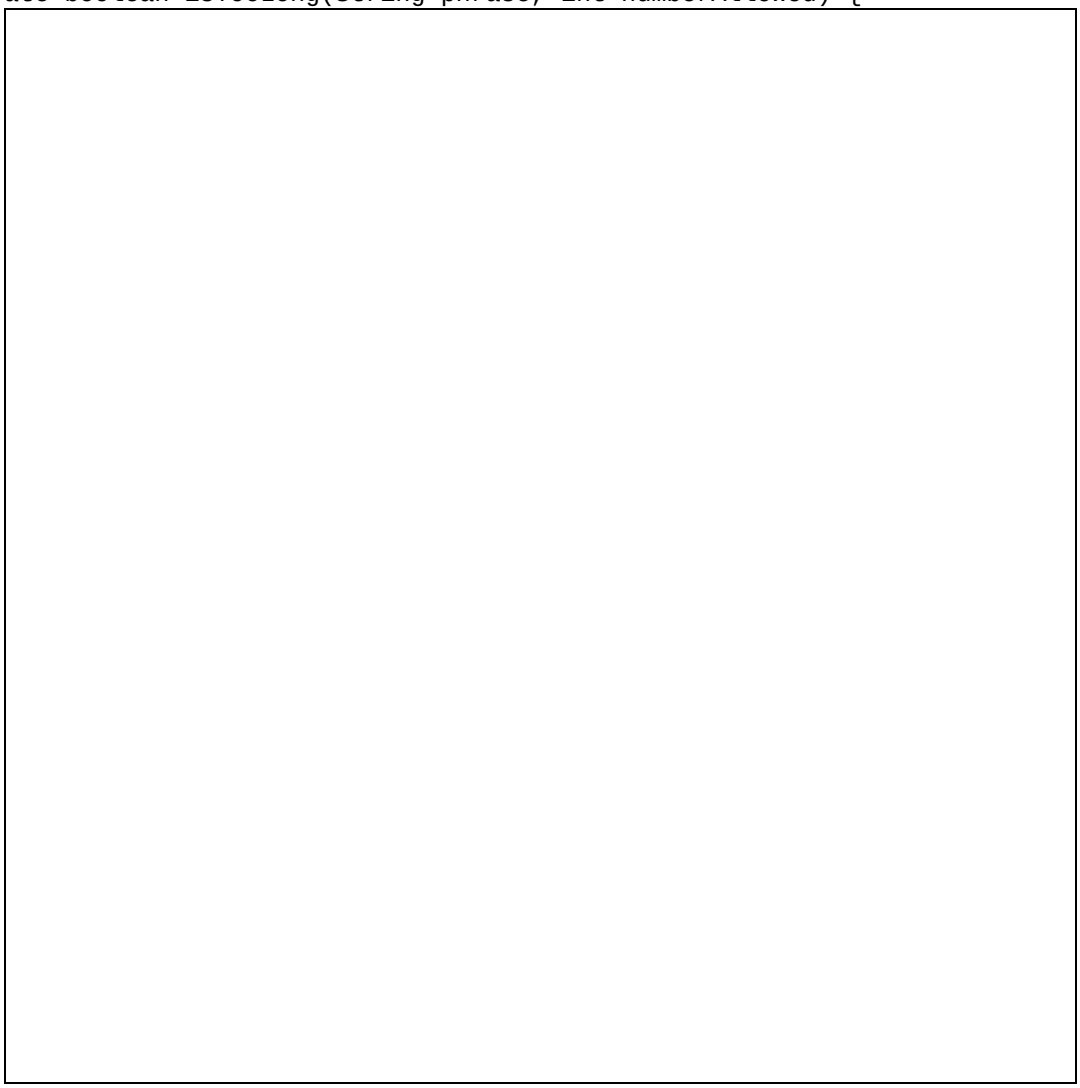
c) Complete the `isTooLong()` method which is passed a String parameter, `phrase`, and an `int` parameter, `numberAllowed`. The method returns `true` if the String parameter contains more characters than the number allowed (given by the parameter, `numberAllowed`), otherwise the method returns `false`. For example, the following code:

```
boolean result = isTooLong("GARBLED", 9);
System.out.println(result);
System.out.println(isTooLong("WHOOPEE DO", 9));
System.out.println(isTooLong("HAPPINESS", 9));
```

would print:

```
false
true
false
```

```
private boolean isTooLong(String phrase, int numberAllowed) {
```



```
}
```

(6 marks)

Question 24 [13 marks]

a) Consider the following Java code:

```
String s = "I like loops";

int i = 0;
while (i < s.length()) {
      System.out.println(s.substring(i));
      i++;
}

System.out.println("I looped " + i + " times");
```

Rewrite this code using a `for`-loop (the first and last lines of code have already been provided for you).

```
String s = "I like loops";









System.out.println("I looped " + i + " times");
```

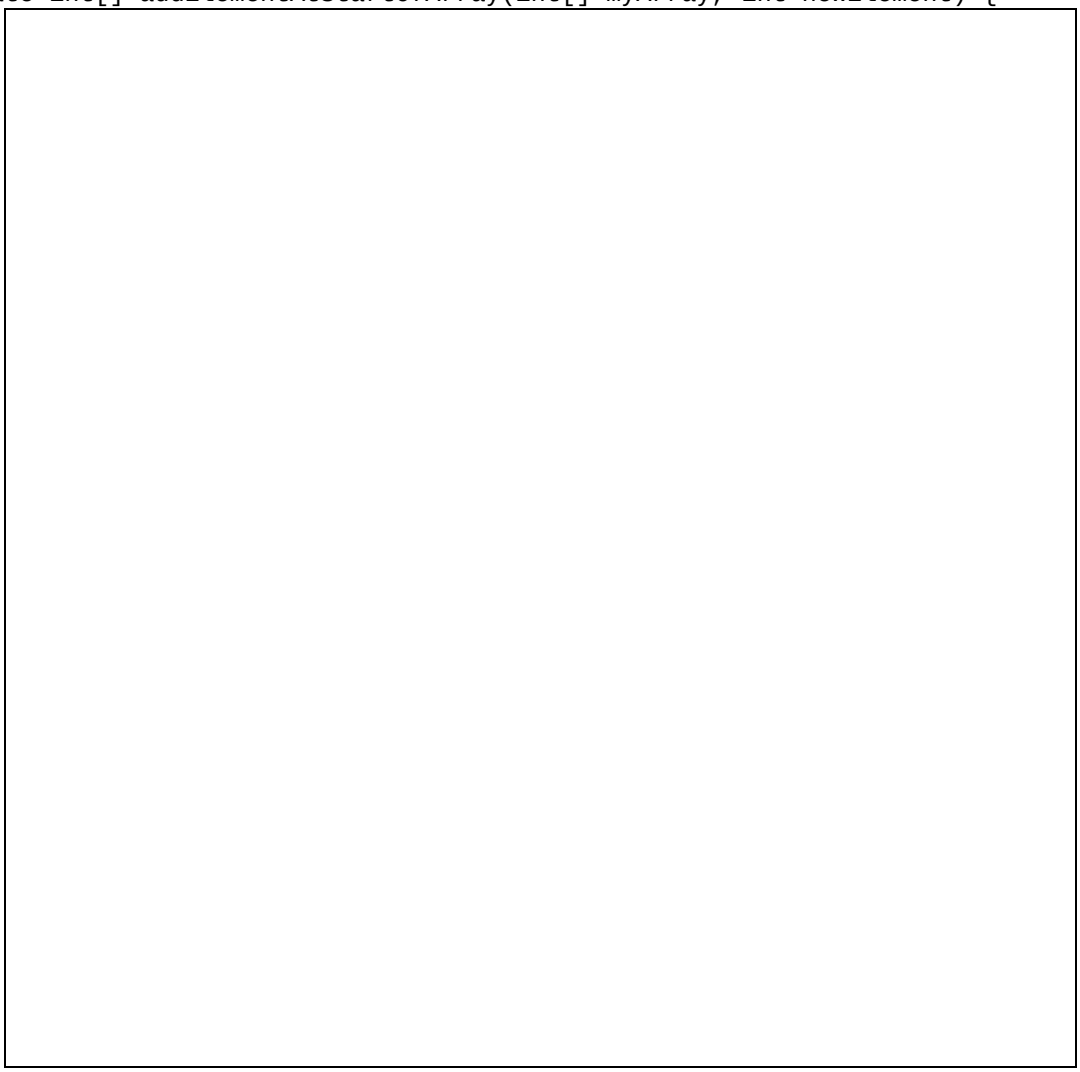
(6 marks)

b) Consider the following Java method. It is passed an array `myArray` and returns an array with all the elements of `myArray` and an additional element `newElement` **at the end of the array**:

```
private int[] addElementAtEndOfArray(int[] myArray, int newElement) {

      int[] biggerArray = new int[myArray.length + 1];

      for (int i = 0; i < myArray.length; i++) {
            biggerArray[i] = myArray[i];
      }

      biggerArray[myArray.length] = newElement;
      return biggerArray;
}
```

Based on the above, write a similar method that adds the new element **at the beginning of the array**, after shifting the index of all other elements by 1. Note: the parameter `myArray` will never be null.

```
private int[] addElementAtStartOfArray(int[] myArray, int newElement) {
```



```
}
```

(7 marks)

Overflow page – please number answers carefully

APPENDIX

String
public int indexOf(char c)
public int indexOf(String string)
public char charAt(int index)
public String substring(int beginIndex, int endIndex)
public int length()
public boolean equals(String comparison)

Math
public static double random()
public static int max(int a, int b)

Integer
public static int parseInt(String s)

Loops
```
for (initialization; loop condition; increment) {
   statement(s)
}

while (expression) {
   statement(s)
}
```