Use the scantron sheet to enter the answer to questions 1 - 100 (pages 1 - 6)

Part I. Mark A for True, B for false. (1 point each)
1. Abstraction allow us to specify an object regardless of how the object will perform its job.
2. Decomposition allow us to deal with complexity.
3. To program means to design, implement and test.
4. A client of a server can use all public features of the server’s specification.
5. Syntax errors are found at compile time.
6. Logical errors are found at run time.
7. Parameters are given initial values by the server.
8. return statements can appear in the implementation of commands.
9. There are as many instance variables as properties in the object.
10. Instance variables are created in the constructor but initialized in the queries and commands.
11. The only difference between a char value and a String value is that the latter can have many characters.
12. A local variable is initialized by the programmer.
13. Client is responsible to fulfill methods post-conditions.
14. A class invariant is a statement about the object’s state.
15. Pre-conditions and post-conditions are not needed to be known when writing a program.
16. A functional test is used to test a complete program while a unit test only tests a component of the program.
17. A data input stream is needed to input data to a program.
18. An instance of java.util.Scanner is an example of an output stream.
19. An interface definition and a class definition differ only slightly.
20. The methods specified in an interface are by default public and abstract.
22. An interface can extend many interfaces while a class can only extend one class.
23. An instance of a subclass can be given as value to a reference variable of the superclass.
24. To overload a method provide more than one specification of the method having different type of parameters.
25. All classes extend the class Object.
26. To overwrite a method in a subclass, the specification of the method appears in a superclass while the subclass only provides the implementation keeping the same specification.
27. An abstract class contains an implementation where all the methods have been declared abstract.
28. Interfaces can be used to defines new Java types.
29. A subclass inherits the superclass’ constructor.
30. You can use composition and inheritance interchangeably.

Part II. Multiple choice. questions. (2 points each)
31. A class definition has information for
   a. the client
   b. the server
   c. both
   d. neither one
   e. one but not the other

32. The following are legal identifiers
   a. X1a
   b. X!a
   c. 3X
   d. #Student
   e. all of the above

33. An object consists of
   a. Properties
   b. queries
   c. commands
   d. state
   e. all of the above

34. Computer Science includes the study of:
   a. design of hardware
   b. design of software
   c. design and study of prog languages
   d. Artificial intelligence
   e. All of the above.
35. An instance variable is labeled:
   a. public
   b. private
   c. either a or b.
   d. neither a nor b.
   e. None of the above.

36. Every java class has:
   a. constructor
   b. queries
   c. commands.
   d. all the above.
   e. At least a and b.

37. This step exposes logical errors that are rarely caught during compilation.
   a. testing
   b. coding
   c. problem analysis
   d. problem statement
   e. all of the above

38. All of these are Java primitive types except:
   a. int
   b. double
   c. single
   d. char
   e. boolean

39. All of these are Java keywords except:
   a. new
   b. package
   c. program
   d. return
   e. else

40. The selection statement is implemented via
   a. return statement.
   b. assignment statement.
   c. while statement.
   d. if-else statement.
   e. select statement.

41. A Java type involves:
   a. all Unicode characters and symbols.
   b. reserved words and identifiers.
   c. character size and font.
   d. syntax rules and related semantics.
   e. a set of values and operations on them.

42. In a client-server relationship between two objects, the following is always true:
   a. The client’s type is known to the server.
   b. The server’s type is known to the client.
   c. The server uses the client’s resources.
   d. The client assumes the server’s identity.
   e. The server creates the client’s instance.

43. Specification(specs) and implementation (impl) relate to each other according to this analogy:
   a. Implementation precedes specification.
   b. They both are one and the same.
   c. Specs is important; implementation is not.
   d. Impl is independent of specs.
   e. Specs sets the job in implementation.

44. An object’s properties are implemented via
   a. private instance variables.
   b. public query methods.
   c. public command methods.
   d. public static final named constants
   e. both a and b.

45. In the hierarchy of Java classes,
   a. All classes are independent.
   b. All classes are related.
   c. Class Object is at the top of the hierarchy.
   d. Class Object is at the bottom of the hierarchy.
   e. Class Math is at the top of the hierarchy.

46. This type of testing considers only expected behavior of a system, not the details of its implementation:
   a. opaque testing
   b. transparent testing
   c. gray-box testing
   d. white-box testing
   e. black-box testing

47. Precondition responsibilities are assigned to ______ while postcondition responsibilities to ______.
   a. server / client
   b. client / server
   c. ensure / require
   d. require / ensure
   e. constructors / methods

48. A subclass inherits
   a. all the methods from superclass
   b. only the queries from superclass.
   c. only the private methods from superclass.
   d. only the commands from superclass.
   e. all the non-public methods from superclass.
49. If a class A implements an interface I
   a. An A instance can be used as an I instance.
   b. A has as many methods as I.
   c. I may have some implementation already.
   d. I can be used as an A instance.
   e. A and I can be used interchangeably.

50. If A is a subclass of class B
   a. An A instance can be used as an instance of B.
   b. B can be used as an instance of A.
   c. A and B can be used interchangeably.
   d. A and B can not be interchanged.
   e. None of the above.

51. Suppose that the code segment in the box below were executed:
   ```java
   int n;
   double x;
   n = 13;
   x = (double) n/3;
   x = 2*x;
   ```
   What would be the value of x after the code segment is executed?
   a. 8.666666666666666
   b. 8.000000000000000
   c. 4.333333333333333
   d. 8
   e. None of these (i.e., all are wrong values).

52. Consider the following class implementation:
   ```java
   class Adder {
       public void addFive( int x ){
           x = x + 5;
       }
   }
   ```
   ```java
   public class AdderTester {
       public static void main( String [] args ) {
           int sum=5;
           sum = sum * 2;
           Adder adder = new Adder();
           int i = 0;
           while ( i<=5) {
               adder.addFive( sum );
               i = i + 2;
           }
           System.out.println( "sum = " + sum);
       } //end main
   } //end class AdderTester
   ```
   What output will be produced when the main method above is invoked? (3 points)
   a. sum = 5
   b. sum = 0
   c. sum = 15
   d. sum = 20
   e. None of these (i.e., all are wrong answers).
Given the variable definitions with initialization:

```java
int i1 = 1;
int i2 = 2;
boolean a = i2 % 2 == 0;
boolean b = i1 % 2 == 0;
boolean c = i2 > 4 % i2;
```

Give the value of each of the boolean expressions in questions 53-60 by marking A for true or B for false. If the expression is not a legal boolean expression, mark C. (1 point each)

53. \((i2 - i1 < 0) \lor (i2 - i1 > 2)\)
54. \(i2/4 \leq i2 - i1\)
55. \(i1 != i2\)
56. \(i1 = i2\)
57. \(a \& \& b\)
58. \(a \lor b\)
59. \(!c \& \& !b\)
60. \(0 < i1 + c\)

For each of the following a, b and c are int variables. Determine what c will contain after each segment of code is executed. Do not be misled by indentation. Choose among the answers below for questions 61-68 (2 points each):

a. 0  
b. 1  
c. 2  
d. 3  
e. None of the above.

61. a = 1;  62. a = 1;  63. c = 1;
b = 2;  c = 1;  if (c == 1)
c = 2;  c = 1;  c = c + 1;
if (a-b > 0)  if (a - b > 0)  if (c == 2)
c = 2;  c = 3;  c = c+1;
else  else  else
c = 3;  c = 0;

64. a = 1;  65. a = 1;  66. a = 2;
b = 2;  b = 2;  c = 1;
c = 1;  c = 0;  if (a > 0)
if (a / b > 0)  while (a < b)  c = 1;
c = 2;  a = a + 1;
else  c = c + b;
c = 3;

67. a = 7;  68. c = 0;
b = 10;  a = 7;
while (a > b &&  b = 10;
b > a){  while (a > b){
    c = c + 1;  c = c + 1;
a = a + 1;  b = b - 1;
    c = c + 1;
}
Assume the class `Rectangle` models a rectangle; its constructor takes integers for length, and width.

```java
Rectangle r1 = new Rectangle( 5, 8);
Rectangle r2 = new Rectangle( 5, 8);
Rectangle r3 = r1;
Rectangle r4 = r2;
```

Mark A for True, B for false for each of questions 80-87. (1 points each)

69. r1 == r2
70. r1.equals(r2)
71. r1 == r3
72. r1 == r4
73. r3.equals(r2)
74. r2 == r4
75. r2.equals(r4)

Mark A for True, B for false for each of the questions 88-100. (1 point each)

76. All classes implement class Object.
77. If A is a subclass of B, A does not have to write a constructor as it can use the one from B.
78. If subclass A implements a constructor, the first statement must be an invocation of its superclass constructor.
79. A subclass A can overwrite any of the methods it inherits from its superclass.
80. An abstract class has some methods labelled abstract.
81. An abstract class does not need to implement all its methods.
82. An interface is like an abstract class that has all its methods labelled abstract and are not implemented.
83. An interface may specify a constructor.
84. A subclass may extends its superclass by adding more instance variables or methods.
85. A superclass’ data can be accessed by any of its subclasses.
86. A subclass instance can access any superclass data that is labelled `protected`.

The composition part of this test starts in the next page. Answer all your questions on those pages. Do not use the scantron sheet for those.
1. Implement the method `getLargest()` in a model class that requests input data from the TUI to find the largest integer entered by the user from a sequence of integers terminated with zero; zero is part of the sequence of numbers used to compute largest. Recall, the model does not directly read the data. When the TUI reads in a number from the user, the TUI uses the model's method `setNextValue`, which sets the next number read. This number is returned by the model's query: `getNextValue`. (10 points)

```java
/**
 * require: an integer number is given.
 * ensure:
 *     this.getNextValue() == number.
 */
public void setNextValue(int number);

public int getLargest(TUI tui) {
```

```
2. Write in the corresponding spaces provided below, preconditions and postconditions for the method nickels, and implementation of the algorithm. (10 points)
Consider the following partially specified class.

```java
/**
 * A DollarChanger models a machine that dispenses a dollar's
 * worth of coins when a dollar bill is inserted.
 */

public class DollarChanger {

    /**
     * The number of nickels required to complete change for a
     * dollar, given that q quarters and d dimes have already
     * been dispensed.
     * For instance, nickels(2,3) would return 4, since 4 nickels
     * (20¢) are required to complete change of a dollar if
     * 2 quarters (50¢) and 3 dimes (30¢) have been dispensed.
     * require:
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * ensure:
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     * *
     */

    public int nickels (int q, int d) {

    }
}
```

} // end of class DollarChanger
3. Implement the queries `letterGrade` and `largestGrade` for the class `Student` is partially specified; an student has 5 numeric grades between 0..100; (12 points)

```java
public class Student {
    private int grade1;
    private int grade2;
    private int grade3;
    private int grade4;
    private int grade5;

    //assume class has public queries for each of the grades.
    /**
     * ensure:
     * returns the letter of the average numeric grade.
     * If letter grade is 'F' and grade5() >= 90, letter grade
     * is 'D'.
     * public char letterGrade() {
```
/**
 * returns largest grade of all student’s 5 grades.
 */
public int largestGrade() { ....}
4. Carefully and completely list all the steps needed in the development of a program. (10 points)

5. Consider the following problem: we need to write a program to maintain checking accounts for Bank customers; assume checking account earns no interest.
   a. Design an object to represent a bank account. (5 points)

   b. Give an example of a state of a bank account based on your design in a. (5 points)
c. Specify and implement the class *BankAccount*. (15 points)
6. Consider the following interface definition modelling a 2-dim shape.

```java
public interface Shape {
    /**
     * returns area of shape.
     */
    public double area();

    /**
     * returns perimeter of shape.
     */
    public double perimeter();

    /**
     * returns diagonal of shape.
     */
    public double diagonal();
}
```

List two shapes for which we can compute each of the queries above. (2 points)

Proceed to take one of the shapes you listed and implement the interface `Shape` for it. (10 points)
7. Consider the following class definition:

```java
public class Circle {
    private int radius;
    public Circle(int radius) {
        this.radius = radius;
    }

    public int area() {
        return Math.PI * radius * radius;
    }

    public int perimeter() {
        return 2 * Math.PI * radius;
    }
}
```

Define a subclass of `Circle` called `CircleWithBorder` (see figure above) which models a circle with a border around it; the client creates an instance by giving both its radius and the thickness of its border. Both the area and the perimeter do not include the border. (10 points)

Assume you want to overload one of the methods of class `Circle`. Where do you need to specify and implement that overloaded method? (2 points)

Assume you want to overwrite one of the methods of class `Circle`. Where do you need to specify and implement the method that overwrites a `Circle`'s method? (2 points)
Overload the constructor for `CircleWithBorder` so that the user need only to give the radius to create one instance giving a default border of 1/100th of the given radius. (5 points)

Overwrite `Circle`'s area method, so that `BorderWithCircle` includes the border as part of its area (5 points).

Given the following declarations:

```java
Circle c1;
CircleWithBorder c2;
```

Choose True (legal) or False (illegal) for each of the following statements (1 point each):

- T  F  `c1 = new CircleWithBorder(5);`
- T  F  `c2 = c1;`
- T  F  `c2 = new CircleWithBorder(10);`
- T  F  `c1 = c2;`
- T  F  `c1 = (Circle) c2;`