University of Calgary
Department of Electrical & Software Engineering
ENSF 480: Principles of Software Design
Practice Exam for Midterm II – Fall 2022

Locations: ICT 102 (Section 01), ICT 121 (Section 02)

Select You Lecture Section:

- **L01 (Mahmood Moussavi)** ☐
- **L02 (Anja Slama)** ☐

Print Your Last Name: _____

Print Your First Name: _____

Print First Letter of You Last Name: _____

Pleased Write Your U of C ID Number on the Last Page (Page 10)

Instructions:

- Please answer Section 1 (multiple-choice questions) on the **scantron answer sheet** and write other answers on the question paper and hand in the question paper and scantron answer sheet when you are done.
- You may **not** use electronic calculators or any electronic devices during the test.
- The test is **closed book.**
- Please print your answers legibly.
- Marks will be deducted for poor programming style such as: lack of clear, consistent indentation, unreasonable choice of variable names, etc.
- Marks will be deducted for unclear diagrams or illegible statements.
- You may use backs of pages for rough work.

NOTE: If you need additional space for questions in sections 2 to 6 you can write on the back of the page, but put a note that code continues on the back of the page

Section I - Multiple Choice Questions

Important Notes:

- Please enter your answers on the scantron answer sheet. Answers on this exam booklet will not be counted.
- If necessary, you may assume that ALL required header files such as <iostream> and the associated namespace are included.

1. Consider the following C++ program:

```
class ExamClass {
public:
    ExamClass(int n);
    ExamClass():lengthM(0), storeM(nullptr) { cout << "0";}
    ~ExamClass();
    ExamClass(const ExamClass& source);
    ExamClass& operator =(const ExamClass& rhs);
private:
    int lengthM;
    int* storeM;
};

ExamClass::ExamClass(int n): storeM(new int[n]), lengthM(n){
    cout << "1";
}

ExamClass::ExamClass(const ExamClass& source):
lengthM(source.lengthM), storeM(new int[source.lengthM+1]){
    for(int i= 0; i < lengthM; i++) storeM[i] = source.storeM[i];
    cout << "2";
}

ExamClass::~ExamClass(){
    delete [] storeM;
    cout << "3";
}

ExamClass& ExamClass::operator =(const ExamClass& rhs){
    if(this == &rhs)
        return *this;
    delete [] storeM;
    lengthM = rhs.lengthM;
    storeM = new int [lengthM];
    for(int i= 0; i < lengthM; i++) storeM[i] = rhs.storeM[i];
    cout << "4";
    return *this;
}

int main(void){
    ExamClass *z;
    ExamClass w (3);
    {
        ExamClass x[1];
        ExamClass t(w);
        z = new ExamClass(2);
        *x = *x;
    }
    delete  z;
    return 0;
}
```

a. 21213333
b. 10013333
c. 10213333
d. 11213333
e. 00213333

2. Which one of the following statements is a correct statement in C++?
   a. An overloaded operator function must have at least a class object as its argument.
   b. An overloaded operator cannot return a pointe, but can return a reference.
   c. All of the above are correct answers
   d. None of the above is a correct answer

3. Class A declares class B as friend. In the above program:
   a. Class A can have access to all data members in class B.
   b. Class B can have access to all data members in class A.

c. Class B can have access to all data members of in class A and class A can have access to data member in class B.
d. None of the above is correct.

4. Assume in a C++ class called Text we want to overload many operators. Which one of the following group of operators must be overloaded as a non-member function?
   a. &&, ||, new
   b. ++, and --
   c. <<, >>
   d. All of the above
   e. None of the above.

5. Which one of the following functions is recommended to be declared virtual :
   a. copy constructors
   b. Assignment operator
   c. Destructor
   d. B and C

6. A UML state transition diagram is normally used to:
   a. Specify the flow of data in system.
   b. Specify the flow of control and sequence of actions in a system
   c. Specify the time-based behaviour of a system or a system's components
   d. All of the above are correct statements

7. Strong coupling leads to a program that:
   a. Is easier to maintain.
   b. Is more efficient.
   c. Is not easier to maintain but it is more efficient.
   d. None of the above

8. A C++ or Java class defines the common characteristics of a group of objects by its. Select the BEST answer:
   a. Identity, attributes, and operations
   b. Attributes, and operations
   c. Identity, behaviour and state
   d. Attributes, operations and relationships
   e. Relationships, operations and multiplicity

9. Object-oriented software development supports two types of hierarchy which includes:
   a. Class hierarchy and entity hierarchy
   b. Whole-part hierarchy and Generalization-Specialization hierarchy
   c. Class hierarchy and structure chart
   d. Class hierarchy and process hierarchy
   e. None of the above

10. A derived class in C++ inherits all member functions except:
    a. Constructors
    b. Copy constructor
    c. Assignment operators
    d. All of the above
    e. None of the above

11. Consider partial definition of the following C++ classes called `Base` and `Child1` and answer the following question:

```
class Base {
public:
    void foo() {cout << "Base foo\n";};
    virtual void bar(){cout << "Base bar\n";}
};
class Child1: virtual public Base {
public:
   void foo() {cout << "Child1 foo\n";};
   void bar() {cout << "Child1 bar\n";}
};
```

The `virtual` keyword in front of `public Base` in class `Child1` means:
   a. Member functions of class `Child1` will be inherited to its subclasses
   a. Member functions derived from `Base` are all `virtual` by default
   b. Both statements (a) and (b) are correct.
   c. None of the above statements about `virtual` keyword in front of `public Base` are correct

12. Which of the following statement(s) is a basic principle for many design patterns
    a. Separation of changeable behaviours from steady behaviours
    b. Programming to interface not to implementation
    c. Programming to implementation not to interface
    d. Both statements (a) and (b) are principle statements for many design pattern
    e. None of the above

13. Java I/O is a real word example of:
    a. Strategy pattern
    b. Decorator pattern
    c. Bridge pattern
    d. Adaptor pattern
    e. None of the above

Consider the partial definition of the following C++ classes, and the given `main` function, to answer the following two questions.

```
class A{
  char * s1;
public:
  A(int n){
    s1= new char[n];
  }
  ~A() { delete [] s1;}
};
```

```
class B : public A {
  char * s2;
public:
  B(int n):A(n) {
     s2=new char[n];
  }
  ~B() { delete [] s2;}
};
```

```
1
2
3
4
5
6
7
8
```

```
int main(void){
  A *p1 = new B(5);
  B *p2 = new B(6);
  p1 = p2;
  p2 = p1;
  // MORE CODE HERE
  return 0;
}
```

14. The above C++ program has a **compilation error** in:
    a. Line 2
    b. Line 3
    c. Line 4
    d. Line 5
    e. Line 6

Consider the partial definition of the following C++ classes, and the given `main` function, to answer the following two questions.

```
class A{
  char * s1;
public:
  A(int n){
    s1= new char[n];
  }
  ~A() { delete [] s1;}
};
```

```
class B : public A {
  char * s2;
public:
  B(int n):A(n) {
     s2=new char[n];
  }
  ~B() { delete [] s2;}
};
```

```
1
2
3
4
5
6
7
8
```

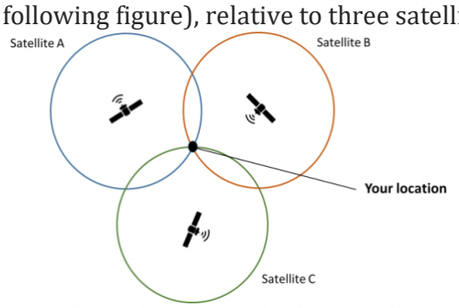
```
int main(void){
  A *p = new B(5);
  // MORE CODE HERE
  Delete p;
  return 0;
}
```

15. There is a serious memory management issue in this program. Which one of the followings addresses the issue:
    a. Constructor of class B is not properly defined
    b. `p is` not properly deleted.
    c. Destructor of class A is not declared virtual
    d. None of the above

16. Boxes on the top of the sequence diagrams represent:
    a. States
    b. Use cases
    c. Messages
    d. Objects
    e. All of the above
    f. None of the above

Section II – Use Case Diagram

In this section you produce a UML design artefacts for a GPS system in car. For your information, the process of calculating a car location is based on ***trilateration*** method that uses overlapping regions to find a common, intersecting area. On a 2-D space consider the location of a receiver on a map (shown as **your-location** in the following figure), relative to three satellites A, B, and C:

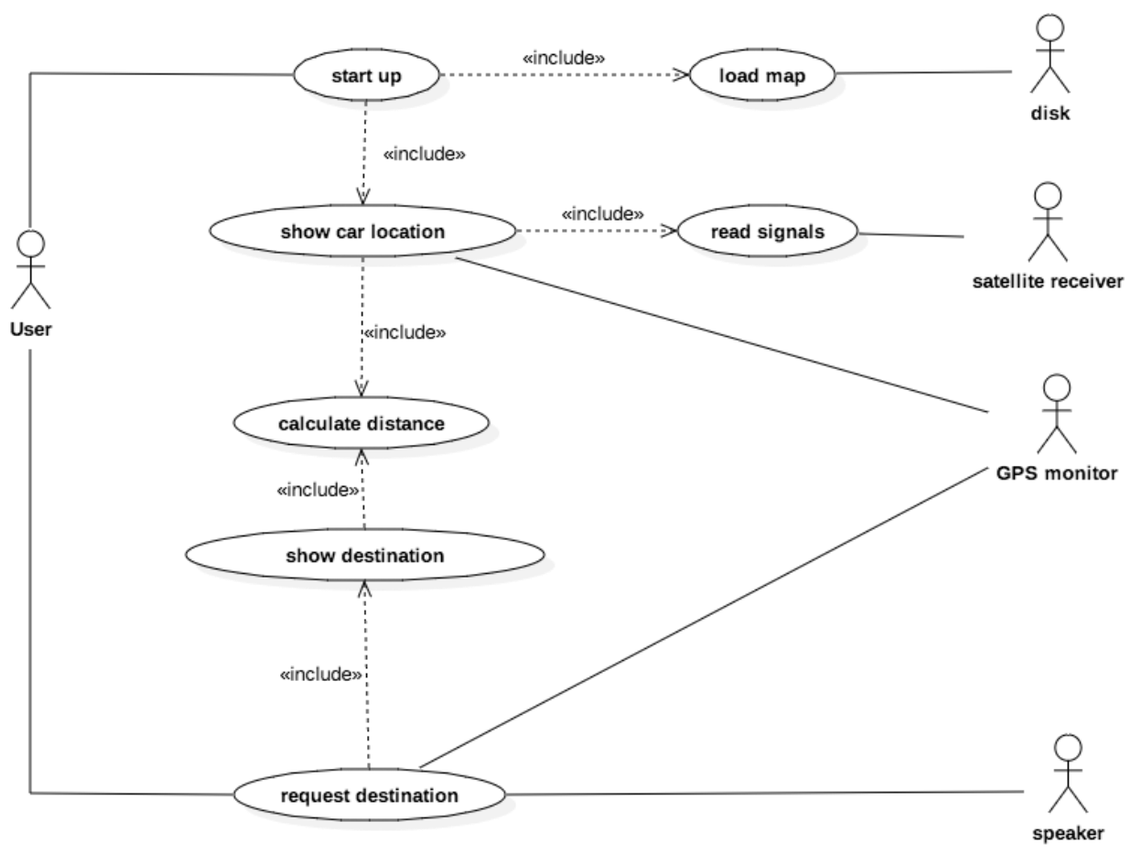


Now if you draw a circle that satellite A is the center and it passes '**your location'** the line that represents the perimeter of the circle also represents all the possible places that have the same distance to the satellite A. If you draw another circle to represent your distance from Satellite B, the number of pointes that can have the same distance to both satellites will reduce to two. And finally, by drawing a third circle, there will be only one point where all three circles intersect and that is your location. The distance between your-location and each satellite is calculated by timing a signal's journey from satellite to receiver. In other words distance is calculated from the time a radio signal travels between satellite and receiver, where the radio signals' travel with speed of light (300,000 km/sec).

Here are some essential required functionalities of this system:

a. An available regional-map (Example: North America) should be loaded from a built-in disk.
b. System should be able to read the signals received from a built-in device called, satellite-receiver.
c. When a radio signal is received, the system should also receive the exact satellite-time (which is the time when the signal was sent), and should be able to covert it to the corresponding local time of GPS location.
d. In concurrence with above item (item c), when a radio signal is received, the system should immediately read the local time.
e. System should be able to calculate its distance from three satellite based on real-time information received in (c), and (d).
f. System should be able to display the current location on the GPS monitor.
g. System should able to provide trip guidance to the user's destination via a built-in speaker and on the GPS monitor

In the following space draw a use case diagram for this system.



Section III – Reverse Engineering from C++ to UML: Consider the following C++ classes:

```
class A: public E{
 protected:
  int id:
 public:
  A(int x);

 void fun1();
 void fun2 ();

};
```

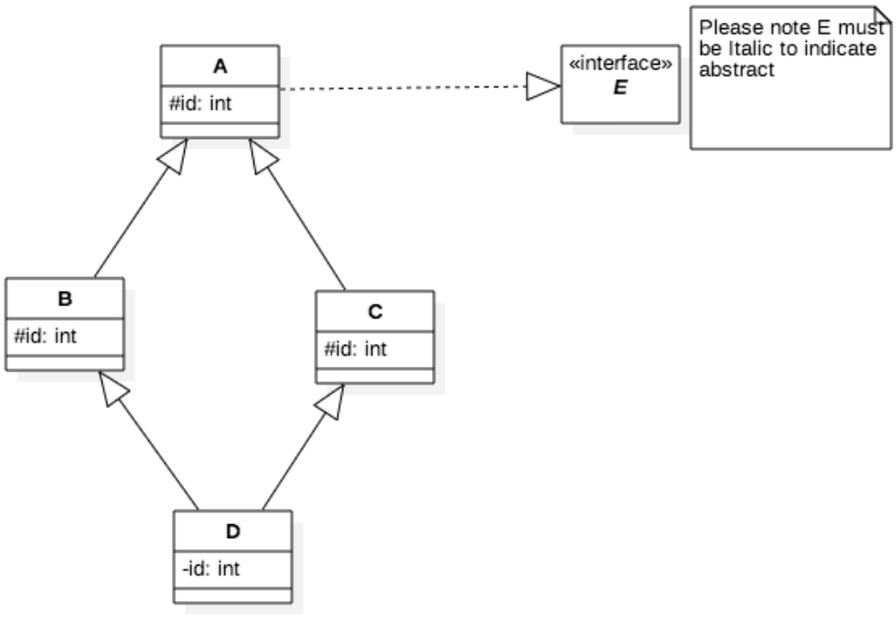
```
class B:public A{
 protected:
  int id;
 public:
  B(int x, int y);
  // more code
 };
```

```
class E {
 public:
  virtual void fun1() =0;
  virtual void fun2() =0;
 // No data members – serves as an interface
};
```

```
class C: public A{
 protected:
  int id;
 public:
  C(int x, int y);
    // more code
 };
```

```
class D:public B,public C{
 private:
  int id;
 public:
  D(int x,int y,int z);
 // more code
};
```

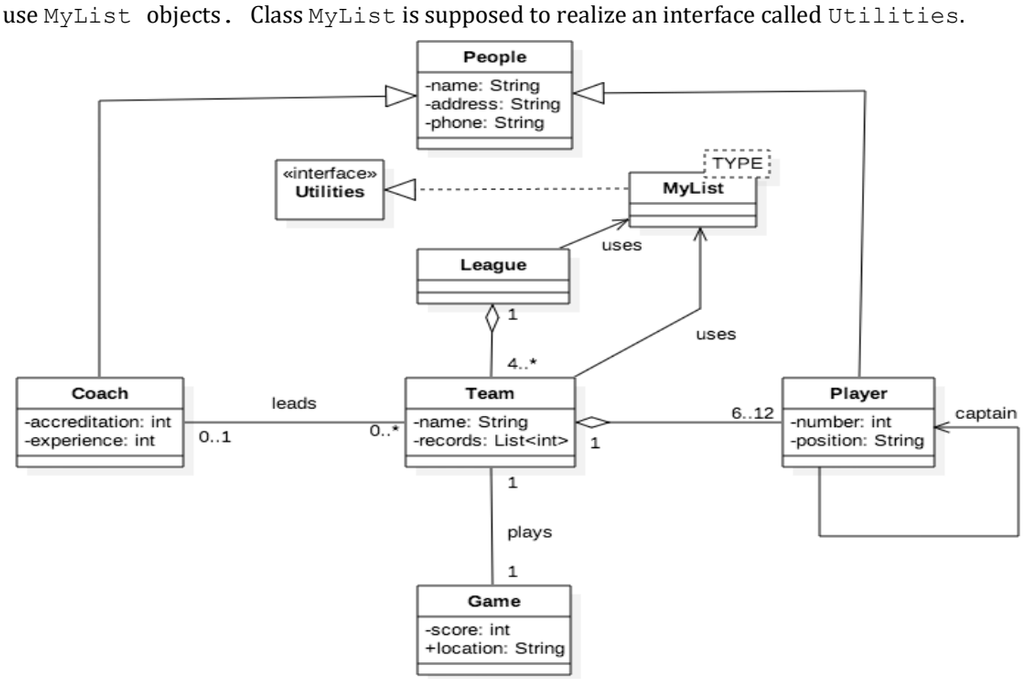
In the following space draw a class diagram that shows the attributes, and relationships such as inheritance and realization among them. Don't worry about behaviours.



SECTION IV – Drawing UML Class Diagram: Draw a UML class diagram (using UML 2.5 notation and syntax), for the following problem statement. For this question you don't have to show class operations, but you should show class attributes.

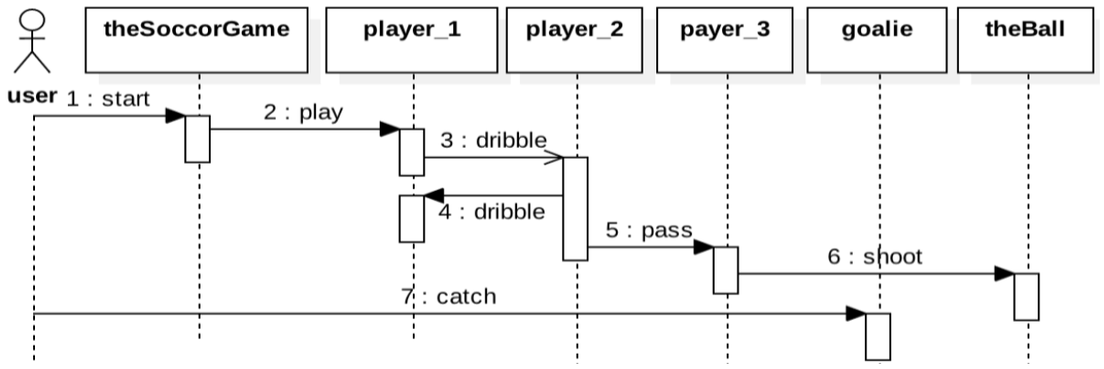
Problem Statement: A Hockey-League is made up of at least four Hockey Teams. Each Hockey-Team has six to twelve players, and one of the players assumes the role of a Captain (*Note: Captain has no additional attribute and no additional functionality – his/her attributes and functionalities are identical to other players*). Each team has a name and a list of its records (an integer `List`). Players have a number (`int` type) and a position (`String` type). Hockey teams play games against each other. Each game has a score and a location. Teams are sometimes lead by a coach. A coach has a level of accreditation and a number of years of experience (both `int` types), and can coach multiple teams. Coaches and players are classified as a person. A person, has names, address, phone number (all `String` types).

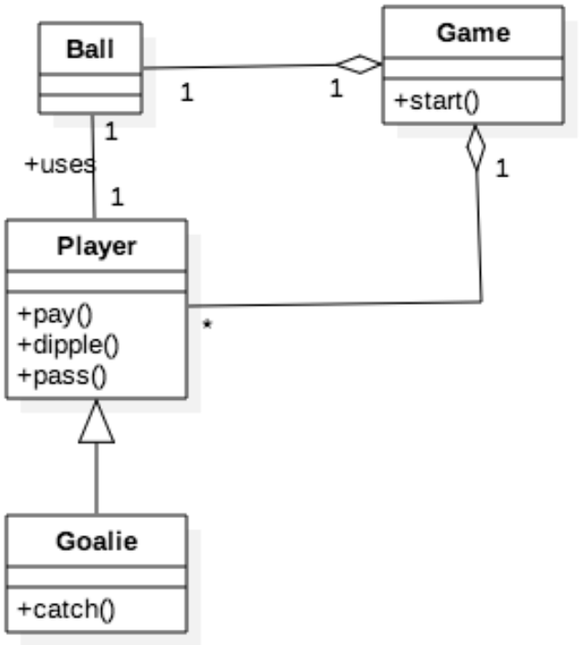
Draw a class diagram based on the given information, and be sure to label all associations with appropriate multiplicities. Apply inheritance, and aggregation (don't worry about composition), or simple association as needed. Also, in your diagram consider and show a generic class called `MyList`. Whenever a class needs to have an aggregation relationship, use `MyList` objects. Class `MyList` is supposed to realize an interface called `Utilities`.



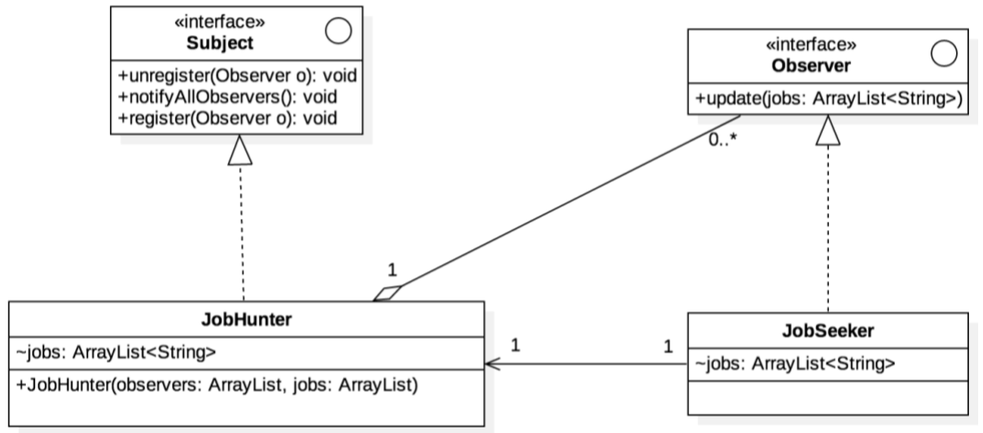
Section V – From UML Interaction to UML Class Diagram:

Consider the following sequence diagram and draw a class diagram the show their relationship and their behaviours. Don't consider user as class and don't worry about attributes of each class.





Section VI – Design Pattern: Based on the following class diagram write the definition of class `JobHunter and JobSeeker` in Java. Please only implement the constructors of these classes, and their methods: `register, notifyAllObservers`, and `update`. You don't have to worry about other functions in these classes.



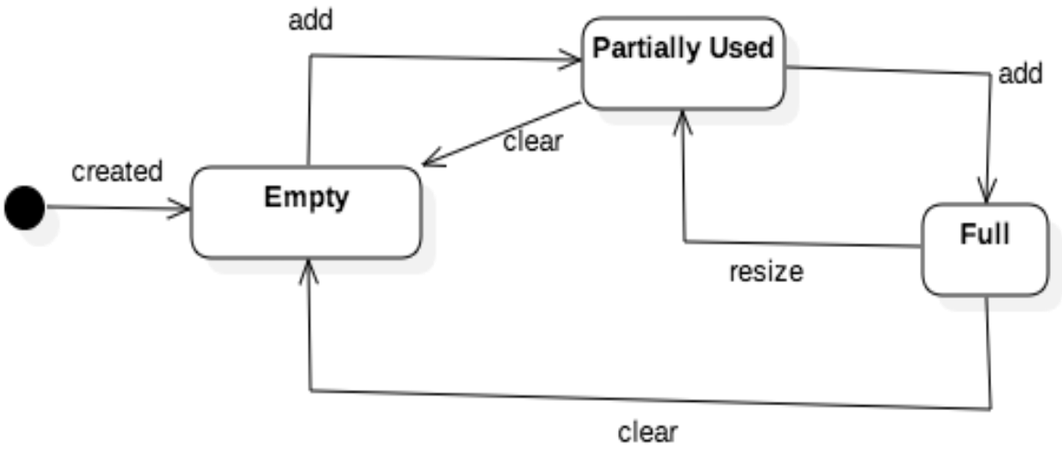
Note: Some of the data members of classes JobHunter and JobSeeker are shown in this diagram. You need to pay attention to relationship among the classes and consider more data members if needed.

Write the definition of your classes in the following space:

```
public class JobHunter implements Subject {
   ArrayList<String> jobs;
   ArrayList<Observer> observers;

   public JobHunter(ArrayList<Observer> o, ArrayList<String> j) {

       jobs = j;
       observers = o;
   }
```

```
   @Override
   public void register(Observer o) {
       observers.add(o);
       o.update(jobs);

   }

   public void notifyAllObservers() {
       for(int i = 0; i < observers.size(); i++){
             Observer o = observers.get(i);
             o.update(jobs);
       }
   }
}


public class JobSeeker implements Observer {
ArrayList <String> jobs;
Subject subject;

   public JobSeeker(Subject s) {
      subject = s;
      subject.register(this);
   }
   @Override
   public void update(ArrayList<String> jobs) {
      this.jobs = jobs;
   }
}
```

Section VII – In this question you should draw a state transition diagram for an object of Java ArrayList. For your information: *In Java when ArrayList is originally created with certain capacity, it is empty and its size is zero. After adding N data its size becomes equal to the capacity and at that point its capacity will be increased (based on certain memory management policy) to allow adding a new data.*
In the following space draw a state transition diagram that expresses the state of an ArrayList object:



Section VIII: An Application **Window** is a graphical user interface to allow users interact with the system or to browse something. In the following space draw a UML state transition diagram that show major states of a window, and the conditions for changing a state to another state.
Here is an example of Window in a PC