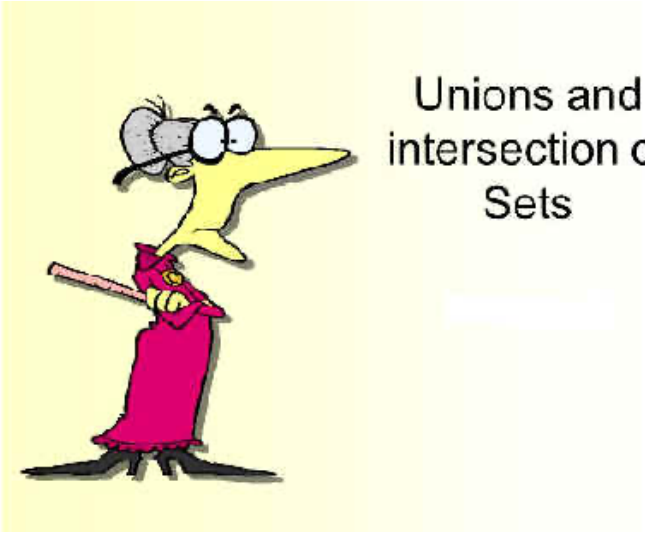
Assignment 6
Due THURSDAY March 11



1. An Integer Set

   a. Define an interface ***SetOperations*** with two methods

      Object union(Object x);
      Object intersection(Object o);

   Notice that the types are Object.
   This is saved as SetOperations.java

   b. Design a class IntegerSet that implements the interface SetOperations.

      public class IntegerSet implements SetOperations

IntegerSet stores a collection of integers such as {3,6,1,2,89,65,74,32,51}.

However, there are **NO DUPLICATES in an IntegerSet**.
You may assume the integers are in the range 0..99.

So the largest IntegerSet is of size 100, since there can be no duplicates

You should use an array of boolean, x[100] to store the numbers in an an IntegerSet.

If num is a a number in the IntegerSet then x[num] is true; if num is not in the set then x[num] is false .

For example if an IntegerSet is {1,3,5,8} then it can be represented by the boolean array partially shown as

F	T	F	T	F	T	F	F	T	ETC

Notice x[1] == true, x[3] == true, x[5] == true and x[8] == true

There should also be a variable that keeps track of the size of a set.

So you should declare an IntegerSet as::

```
public class IntegerSet implements SetOperations
{

    private boolean[] set;
    private int size;
    etc

}
```

Constructors:

There should be

- one default constructor that creates an empty set -- everything in set is false
- a two argument constructor that accepts an array of int along with the size of the array
  and creates an IntegerSet.

  Note: the array may have duplicates but the IntegerSet cannot.
  So if s is an array with 9 values, i.e. s= {1,2,3,4,5,5,5,5,5} then the
  constructor creates the integer set {1,2,3,4,5} (no duplicates) and the
  size is 5. Again, an Integer set has no duplicates.
- There should be a two argument constructor that accepts a boolean array bool and a size of the integer that it stores. This is simple:

```
public IntegerSet(boolean[] set, int size|)
 {
    this.set = set;
    this.size = size;
 }
```

Methods of IntegerSet should include:

```
int getSize();
boolean elementOf(int x);      // is x a member of the set? true or false
void add(int x)                // adds x to a set, a set has no duplicate items
```

IntegerSet should override the methods of Object:

boolean equals(Object o) and
String to String()

Note:

- Two sets are equal if they have the same elements

- toString() returns a string with all set elements such that there is a single space between two elements

  For example, if an IntegerSet is { 1,2,3,4,5}
  toString() returns "1 2 3 4 5"

Define the two methods of the interface:

Object union(Object x)

If x and y are IntegerSet objects then x.union(y) returns an IntegerSet, z, containing the integers in at least one of the two sets, Of course, set z contains no duplicates. For example if x = { 1,2,3,4,5} and y = { 3,4,5,6,7,8} then z = {1,2,3,4,5,6,7,8}. The union operation just merges the sets eliminating duplicates, that is union combines two sets. Hint: use the "or" (||) operator

Object intersection(Object x)

If x and y are IntegerSet objects then x.intersection(y) returns an IntegerSet, z, containing the integers in both x and y. Of course, set z contains no duplicates. For example if x = { 1,2,3,4,5,6,9} and y = { 3,4,5,7,8,9} then z = {3,4,5,9}. The intersection operation finds elements common to both sets.
Hint: Use the "and" (&&) operator.

Test you class with the following class

```
public class TestIntegerSet
{

   public static void main(String[] args)
   {
      int[] list = {1,2,3,4,5,5,5};
      IntegerSet s = new IntegerSet(list,7);
      IntegerSet t = new IntegerSet(); //empty set

      for (int i = 10; i >= 3; i--)
```

```
            t.add(i);

        System.out.println("The elements of set s are ");
        System.out.println(s);
        System.out.println();
        System.out.println("The elements of set t are ");
        System.out.println(t);
        System.out.println();

        System.out.println("3 is an element of s "+ s.elementOf(3));
        System.out.println("9 is an element of s "+ s.elementOf(9));


        Object set1; //  notice the type is Object
        set1 = s.union(t);
        System.out.println("The union of s and t : ");
        System.out.println(((IntegerSet)set1)); // notice the downcast


        int[] a1 = { 1,2,3,4,5,6,6,2};
        int[] b2 = { 2,3,6,3};
        IntegerSet a = new IntegerSet(a1,8);
        IntegerSet b = new IntegerSet(b2,4);
        System.out.println("Set a : "+ a);
        System.out.println("Set b : "+ b);

        Object set2;
        set2 = a.intersection(b);
        System.out.println("The intersection of a and b : ");
        System.out.println(((IntegerSet)set2));

        int [] list1 = {1,2,3,4,5,5};
        IntegerSet w = new IntegerSet(list1,6);
        System.out.println("s equals w: " +s.equals(w));
        System.out.println("s equals t: " +s.equals(t));
    }
}
```

c. A particular lottery allows people to play any set of numbers from 0 through 99. Each number played costs $100. There is one winning number chosen each week. A group of friends play the lottery and each one has some set of favorite numbers. Possibly, some of the friends have chosen the same numbers. They decide to pool their numbers and split the winnings if any one of their numbers wins.

Write a test class, Lottery, that creates three IntegerSet objects, containing the lottery numbers played by three different friends. Print each of these sets. Your test class should create a merged set from the three sets and print out all the numbers in it, and how much it will cost to play these numbers (i.e., how many numbers). Remember union returns Object so you may have to downcast as in the main(...) method of part (b).

2. A Student class

a. Here is the binary search method from CS 103 . This method searches a sorted array of **integers**.

```
public static int search(int[] x , int n, int key)
// pre-condition: x is a sorted array of n integers; key has an integer value
//                    x is sorted in ascending order;
// returns:  the position of key in x or -1 if key is not found

{

   int lo = 0;                  // lowest index of the array
   int  hi = n-1;               // highest index
   int  mid;                    // middle index

    while (hi >= lo)
   {
```

```
        mid  = (hi + lo)/2;       // get the middle index
        if (key == x[mid])
            return mid;           //key found --exit
        if (key < x[mid])
           hi = mid - 1;         //eliminate x[mid] thru x[hi]
        else
             lo = mid + 1;        // eliminate x[lo] thru x[mid]
    }

    return -1;                 // key not found
}
```

Implement a utility class BinarySearch with one static method that can perform a binary search on a sorted array of any object type that implements Comparable. Remember to search an array with binary search, the array must be sorted. Look at what I did with selection sort. The red statements need to be changed.

This is very simple and you need only change a few statements. You are now sorting Comparable objects not ints.

b. Implement an abstract class Person has the following fields

String lastName
String firstName
String SSNumber

Include a default constructor, a three argument constructor, and getter and setter methods.

Person implements Comparable based on the String lastName+firstName. (remember String has a compareTo() method)

Person also implements
boolean equals(Object o)

based on lastName + firstname

c. A Student class extends Person (a Student *is-a* Person) and has the additional fields

double GPA
String campusAddress;

Student has the appropriate constructors.

Student implements

String toString(), (inherited from Object)

which returns all the information about the student

d. Another class Roster has as data

an array of Student[], (assume at most 35 students)
the number of students in the class (int)

The constructors are

- a default constructor that makes an empty roster
- a constructor that reads a text file such that
  each line of the file consists of:

  lastName firstName SSNumber GPA campusAddress

The methods are:

- void sort()
  // sorts by lastName + " "+ firstName, you can just call the generic/utility sort from class (SelectionSort.sort(...))

- int search (String lastname, String firstname) // **calls the static binary search**

The class calls the binary search to find the student and prints that student's data (uses toString()) or
a message that the student is not in the class

Hint: The search method of the class accepts the last name and first name of a student.

However, the utility binary search method accepts an object as the key. So you will have to pass a Student object to the static binary search method. You can use dummy fields for SSNumber, gpa, and campusAddress:

```
Student key = new Student(last, first, "",0.0,"");
```

and pass key to the binary search method (which searches on name only).

We did this with LinearSearch

- int addStudent(Student s) // make sure to sort after an addition
- printClass() //prints all data for each student in the class,
- void menu()
  // prompts for a string

  "S" search
  "A" add a student
  "P" print the class roster
  "E" exit

- void performAction(String choice)
  // performs the action indicated by choice

this routine works as:

```
if (choice.equals("A"))
{
  // ask for the data for a student, last name, first name, gpa etc
  // create a student object s
 // call add(s);
}
```

```
else if (choice.equals("S"))
{
 // ask for the last name and first name
 // pass this to search and search() returns an int ( -1 if not found)
// the int returned is the place in the array, so print the data or say "not found"
}

else if (choice.equals("P")
// print all the data in the array
```

Include a main method in Roster:

```
public static void  main(String [] args)
{
   Roster r = new Roster("students.txt");
   String choice;
   do
   {
      System.out.println();
      choice = r.menu();
      r.performAction(choice);
      System.out.println();
      } while (!choice.equals("E"));
}
```

Here is data for students.txt. I have also emailed you thefile.

```
HOWARD CURLY 123456789 2.6 OHARA
PARKER PETER 345676567 3.2 HOLYCROSS
GOOCHE AGNES 111456789 3.2 BOLAND
BOND JAMES 234876789 3.9  BOLAND
HOWARD MOE 222333444 2.9 SULLIVAN
SIMPSON MARGE 123678989 2.5 VILLATERESA
BENIS ELAINE 987678765 3.6 BOLAND
KRAMER COSMO 123445435 2.2 OHARA
ROSE JOHNNY 234765876 3.3 CORR
ROSE MOIRA 333444555 3.8 BOLAND
ROSE DAVID 986787565 2.3 CORR
ROSE ALEXIS 345678903 3.1 OHARA
SCHITT ROLAND 234657898 3.9 SULLIVAN
SCHITT JOCELYN 986867564 3.1 VILLATERESA
BUDD STEVIE 587857675 2.6 HOLYCROSS
```

SIMPSON LISA 909898979 3.9 BOLAND
SIMPSON BART 234444443 1.0 OHARA
SANDS TWYLA 285655345 3.4 CORR

All data is stored in upper case. However, a user need not enter the data in upper case, so use the method toUpperCase() from the String class to convert all user data to upper case.

Here is typical output. User input is in red.

S : search for a student
A : add a student
P : print all student
E : exit
Choice: S
Last name: Simpson
First name: Bart
SIMPSON BART 234444443 1.0 OHARA

S : search for a student
A : add a student
P : print all student
E : exit
Choice: A
Last name: GRIFFIN
First name: STEWIE
SS Number: 34567129
GPA: 4.0
Campus Address: BOLAND

S : search for a student
A : add a student
P : print all student
E : exit
Choice: S
Last name: GRIFFIN
First name: STEWIE

GRIFFIN STEWIE 34567129 4.0 BOLAND

S : search for a student
A : add a student
P : print all student
E : exit
Choice: S
Last name: D
First name: PAULY
D PAULY: not found

S : search for a student
A : add a student
P : print all student
E : exit
Choice: E
Bye