CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce

343 Davis Hall

alphonce@buffalo.edu

Office hours:

Tuesday 10:00 AM – 12:00 PM*

Wednesday 4:00 PM – 5:00 PM

Friday 11:00 AM – 12:00 PM

OR request appointment via e-mail

**Tuesday adjustments: 11:00 AM – 1:00 PM on 12/6*

ROADMAP

Last time

- wrapper classes
- exercise 07

Today

- equality testing
- exercise 08

Coming up

- exercises 09 and 10
- search (linear and binary)

ANNOUNCEMENTS

FINAL EXAM NOTICE

CHECK YOUR ENTIRE FINAL EXAM SCHEDULE!

http://blogs.advising.buffalo.edu/beadvised/posts/have-you-checked-your-final-exam-schedule-4/

Room assignments will be announced at a later date.

Lab 11

Due 9:00 PM on last day of classes for everyone.

This week – regular recitations and UTA office hours.

Next week – recitations are office hours.

Required functionality 80 pts.
Optional functionality – pick and choose.

100 points is full credit.
You can get more than 100 points.

Lecture exercises will give hints for lab 11 functionality.

EQUALITY TESTING

Equality testing

Equality testing:

- of **primitive (integral/boolean) values**: ==, as in x==y
- be careful doing == with **floating point** numbers!
  - rather than testing (x == y), test Math.abs(x-y)<epsilon
- of **objects**: equals method, as in x.equals(y)

Consider:

```
Color a = new Color(255,0,0);
Color b = new Color(255,0,0);
Color c = a;
```

what is value of	what is value of
(a==b)	(a==c)
a.equals(b)	a.equals(c)

equals method

- override when defining a new class
- must be consistent with the compareTo method for types that can be ordered

EXERCISE 07
solution

EXERCISE 07

Define a method with this header in a class named quiz.Question:

```
public HashSet<HashSet<Integer>> answer(ArrayList<String> list)
```

Define the method so that it returns a partition of list in which each subset consists of contiguous positions which have the same value in list.

Examples

null partitions into {}

“” partitions into {}

“aaa” partitions into {{0,1,2}}

“aab” partitions into {{0},{1,2}}

“aabbccaaa” partitions into {{0,1},{2,3},{4,5},{6,7,8}}

```
public HashSet<HashSet<Integer>> answer(ArrayList<String> list) {

}
```

```
public HashSet<HashSet<Integer>> answer(ArrayList<String> list) {

  HashSet<HashSet<Integer>> partition = new HashSet<HashSet<Integer>>();









  return partition;

}
```

```
public HashSet<HashSet<Integer>> answer(ArrayList<String> list) {

  HashSet<HashSet<Integer>> partition = new HashSet<HashSet<Integer>>();
  if (list != null && !list.isEmpty()) {









  }
  return partition;

}
```

```
public HashSet<HashSet<Integer>> answer(ArrayList<String> list) {

  HashSet<HashSet<Integer>> partition = new HashSet<HashSet<Integer>>();
  if (list != null && !list.isEmpty()) {
    HashSet<Integer> match = new HashSet<Integer>();
    partition.add(match);
    match.add(0);
    for (int i=1; i<list.size(); i=i+1) {









    }
  }
  return partition;

}
```

```
public HashSet<HashSet<Integer>> answer(ArrayList<String> list) {

  HashSet<HashSet<Integer>> partition = new HashSet<HashSet<Integer>>();
  if (list != null && !list.isEmpty()) {
    HashSet<Integer> match = new HashSet<Integer>();
    partition.add(match);
    match.add(0);
    for (int i=1; i<list.size(); i=i+1) {
      if ( list.get(i).equals(list.get(i-1)) ) {
        match.add(i);
      }









    }
  }
  return partition;

}
```

```
public HashSet<HashSet<Integer>> answer(ArrayList<String> list) {

  HashSet<HashSet<Integer>> partition = new HashSet<HashSet<Integer>>();
  if (list != null && !list.isEmpty()) {
    HashSet<Integer> match = new HashSet<Integer>();
    partition.add(match);
    match.add(0);
    for (int i=1; i<list.size(); i=i+1) {
      if ( list.get(i).equals(list.get(i-1)) ) {
        match.add(i);
      }
      else {
        match = new HashSet<Integer>();
        partition.add(match);
        match.add(i);
      }
    }
  }
  return partition;

}
```

EXERCISE 08

EXERCISE 08

Finding matches in 2D

(example & process on board)

EXERCISE 08

Define a method with this header in a class named quiz.Question:

```
public HashSet<Point> answer(Point p, ArrayList<ArrayList<String>>
board)
```

Define the method so that it returns a HashSet containing those points on the board whose positions are adjacent to p and have the same contents.

Consider this board

```
aabbc
abbcd
ddddd
efffd
```

Answer for null is { }

Answer for (0,0) is { (1,0) , (0,1) }

Answer for (2,4) is { (1,4) , (2,3) , (3,4) }