CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce
343 Davis Hall
alphonce@buffalo.edu

Office hours:
Thursday 12:00 PM – 2:00 PM
Friday 8:30 AM – 10:30 AM
OR request appointment via e-mail
Turn off and put away electronics:

- cell phones
- pagers
- laptops
- tablets
- etc.
Where we’ve been
control structures

Today
collections

Where we’re heading
collections
search
I have unclaimed exams with me.

Pick up at end of class.

Grading questions? Come to office hours.
CONTROL STRUCTURES
if ( <expr> ) <stmt_1> else <stmt_2>
if ( <expr> ) <stmt>
while ( <expr> ) <stmt>
public void message() {
    System.out.print("I ");
    int timesAlreadyPrinted = 0;
    while (timesAlreadyPrinted < 3) {
        System.out.print("really ");
        timesAlreadyPrinted = timesAlreadyPrinted + 1;
    }
    System.out.println(" like spring break!");
}
Why should the method be so rigid?

Why must it always print ‘really’ three times?
public void message() {
    System.out.print("I ");
    int timesAlreadyPrinted = 0;
    while (timesAlreadyPrinted < 3) {
        System.out.print("really ");
        timesAlreadyPrinted = timesAlreadyPrinted + 1;
    }
    System.out.println(" like spring break!");
}
public void message(int totalNumberOfTimes) {
    System.out.print("I ");
    int timesAlreadyPrinted = 0;
    while (timesAlreadyPrinted < totalNumberOfTimes) {
        System.out.print("really ");
        timesAlreadyPrinted = timesAlreadyPrinted + 1;
    }
    System.out.println(" like spring break!");
}
COLLECTIONS
A collection object can store arbitrarily many (references to) objects.

We will first learn to become users/clients of collections.

Next semester we will learn to become builders of collections.
All collection classes in Java are subtypes of the `java.util.Collection<E>` interface.

`<E>` is new syntax

E is a type variable, and denote the element type of the collection:

- `Collection<String>` denotes a collection of String objects
- `Collection<ActionListener>` denotes a collection of ActionListener objects
Among the methods specified in the interface:

- **boolean add(E item)** --- tries to add item to the collection; if this is successful, true is returned, false otherwise

- **boolean remove(Object item)** --- tries to remove (one occurrence of) item from the collection; if this is successful, true is returned, false otherwise

- **boolean contains(Object item)** --- returns true if item is in the collection, false otherwise

- **int size()** --- return the number of items currently in the collection
Two specific collections
(defined in java.util)

ArrayList<E>
- permits duplicates
- allows client to control order of elements

HashSet<E>
- does not permit duplicates
- does not allow client to control order of elements
To declare a variable of type HashSet of String:

    HashSet<String> names;

To create a HashSet of String object, and assign its reference to the variable declared above:

    names = new HashSet<String>();
channel 1
How would you declare ‘x’ to be a variable whose type was ArrayList of ActionListener?

A. ArrayList<ActionListener> x;
B. ActionListener<ArrayList> x;
C. ActionListener[] x;
D. ArrayList x = new ActionListener();
How would you declare ‘x’ to be a variable whose type was ArrayList of ActionListener?

A. `ArrayList<ActionListener> x;`
B. `ActionListener<ArrayList> x;`
C. `ActionListener[] x;`
D. `ArrayList x = new ActionListener();`

Convince your neighbor your answer is correct.
How would you declare ‘x’ to be a variable whose type was ArrayList of ActionListener?

A. ArrayList<ActionListener> x;  ✔
B. ActionListener<ArrayList> x;
C. ActionListener[] x;
D. ArrayList x = new ActionListener();
To add a String to the HashSet:

```java
names.add("Fred");
```

To remove a String from the HashSet:

```java
names.remove("Fred");
```
channel 1
How many Strings will be contained in the HashSet<String> x?

```java
x = new HashSet<String>();
x.add("Fred");
x.add("Wilma");
x.add("Fred");
x.remove("Fred");
```

A. 1  
B. 2  
C. 3  
D. 4
How many Strings will be contained in the HashSet<String> x?

x = new HashSet<String>();
x.add("Fred");
x.add("Wilma");
x.add("Fred");
x.remove("Fred");

A. 1  
B. 2  
C. 3  
D. 4
How many Strings will be contained in the HashSet<String> x?

```java
x = new HashSet<String>();
x.add("Fred");
x.add("Wilma");
x.add("Fred");
x.remove("Fred");
```

A. 1  
B. 2  
C. 3  
D. 4
channel 1
How many Strings will be contained in the ArrayList<String> x?

```java
x = new ArrayList<String>();
x.add("Fred");
x.add("Wilma");
x.add("Fred");
x.remove("Fred");
```

A. 1
B. 2
C. 3
D. 4
How many Strings will be contained in the ArrayList<String> x?

x = new ArrayList<String>();
x.add(“Fred”);
x.add(“Wilma”);
x.add(“Fred”);
x.remove(“Fred”);

A. 1
B. 2
C. 3
D. 4

Convince your neighbor your answer is correct.
How many Strings will be contained in the ArrayList<String> x?

```java
x = new ArrayList<String>();
x.add("Fred");
x.add("Wilma");
x.add("Fred");
x.remove("Fred");
```

A. 1  
B. 2  
C. 3  
D. 4
Let’s try out some examples.