We’re in the middle of the quarter! Let’s review what we (YOU but ME AS WELL!!) can improve.
What you like/dislike in/from….
• PSAs
• Group discussions/clicker questions (including the fact that it’s required)
• Lectures
• Lecture slides
• Starting PSAs early
• Tutors

ArrayLists (the array that never ends)
1. Where is the origin located for each component in the Java graphics coordinate system?

A. Lower left corner

**B. Upper left corner**

C. Lower right corner

D. Upper right corner
2. Write a line of Java code for drawing the outline of a circle centered on location (4,4) with radius 2. (Remember you may want to check the description of the method to do this in the official documentation)

```
drawOval(int x, int y, int w, int h)
```

A. `g.fillOval(4,4,2,2);`
B. `g.fillOval(2,2,4,4);`
C. `g.drawOval(4,4,2,2);`
D. `g.drawOval(2,2,4,4);`
3. Which Java package does the ArrayList class belong to?

A. java.awt
B. java.lang
C. java.io
D. java.util
4. How do you determine the number of elements in an ArrayList object named `myList`?

A. `myList.length`

B. `myList.get(-1)`

C. `myList.size()`

D. `myList.clear()`
Midterm

• Again nice job! (better than you thought)
  – Mean = 7.7, Median = 8
  – 21 perfect scores!
  – BUT: 45 scores < 6. Please come see me!
But sometimes an exact type is not known until **run-time:**

- The compiler will assume the object is of the *declared* type.

The constructor still determines the *actual* type of the Object.

- At run-time, Java will use the *actual type's* latest (most-derived) methods.

A: `Person p = new Student( "Sally", 16 );`
B: `System.out.println("p's name is " + p.name);`
C: `System.out.println( p.isAsleep( 24 ) );`
D: `p.status( 24 );`
E: `System.out.println("P is taking " + p.units);`

One of these lines of code will cause an error. Which one? (And how do you fix it)
Casting!

But sometimes an exact type is not known until **run-time**:  
- The compiler will assume the object is of the **declared** type.

The constructor still determines the **actual** type of the Object.  
- At run-time, Java will use the **actual type's** latest (**most-derived**) methods.

A: `Person p = new Student( "Sally", 16 );`
B: `System.out.println("p's name is " + p.name);`
C: `System.out.println( p.isAsleep( 24 ) );`
D: `p.status( 24 );`
E: `System.out.println("P is taking " + ((Student)p).units);`

Casting changes the type of the reference to the object to make the compiler believe us that the object can really do what we ask it.  
Casting can be dangerous, so be sure you know what you’re doing!
Casting changes the type of the reference
(NOT THE OBJECT)

Person p = new Student( "Sally");
Casting changes the type of the reference (NOT THE OBJECT)

“Trust me, this object is actually a Person”

Person p = (Person)(new Student( "Sally"));

The cast makes the object *look* like a plain Person at compile time. But the object is actually still a Student.
Casting changes the type of the reference (NOT THE OBJECT)

Person p = (Person)(new Student("Sally"));

The cast makes the object *look* like a plain Person at compile time. But the object is actually still a Student.

At runtime, Java “applies the cast” to the object, which does not change the type of the object, but will result in an error if the types don’t match.
Casting changes the type of the reference (NOT THE OBJECT)

Person \( p = \) new Person( “Sally” );
Student \( s = p; \)

Will this code cause an error?

A. Yes, at compile time  B. Yes, at run time  C. No

Types don’t match!
Casting changes the type of the reference (NOT THE OBJECT)

Person \( p = \text{new Person( "Sally" )}; \)
Student \( s = (\text{Student})p; \)

Will this code cause an error?
A. Yes, at compile time
B. Yes, at run time
C. No

ClassCast Exception! (we lied ... there is no Student there ... )
ArrayLists

```java
public void setUniqueAndCounts( String[] words )
{
    int index = 0;
    int uniqueIndex = 0;
    int numUniqueWords = countUnique( words );
    uniqueWords = new String[numUniqueWords];
    counts = new int[numUniqueWords];
    while ( index < words.length )
    {
        int count = getCount( words, index );
        uniqueWords[uniqueIndex] = words[index];
        counts[uniqueIndex] = count;
        index += count;
        uniqueIndex++;
    }
}
```

Why did we have to count the number of unique words before filling in the uniqueWords and counts arrays in PSA3?
A. We didn’t. This is just better style.
B. Arrays are fixed size, so we had to know how big to make them
C. We needed the number of unique words to control the while loop
If we would use ArrayLists...

```java
public ArrayList counts;
public ArrayList uniqueWords;

public void setUniqueAndCounts( String[] words )
{
    uniqueWords = new ArrayList();
    counts = new ArrayList();
    int index = 0;
    int uniqueIndex = 0;
    while ( index < words.length )
    {
        int count = getCount( words, index );
        uniqueWords.add(words[index]);
        counts.add( new Integer( count ) );
        index += count;
    }
}
```

So we need a Wrapper class
(An object that holds a primitive value)
Java will do this automatically if you add an int to an ArrayList

ArrayLists only hold Objects, NOT PRIMITIVES
Another Use for Casting: ArrayLists

```java
public void displayWords()
{
    JFrame wordFrame = new JFrame("Word cloud");
    wordFrame.setLayout(new FlowLayout(FlowLayout.LEFT));

    for ( int i = 0; i < uniqueWords.size(); i++ ){
        int fontScale = 2 * counts.get( i );

        // More code not shown
    }
}
```

The last line above has an error. Why?

A. get is not the right method to get an element from an ArrayList
B. counts.get(i) will return an Integer, which cannot be multiplied by 2
C. counts.get(i) returns a reference to an Integer, which when multiplied by 2 gives another Integer, and not an int
D. counts.get(i) will return a reference to an Object, which cannot be multiplied by 2
Another Use for Casting: ArrayLists

public void displayWords()
{
    JFrame wordFrame = new JFrame("Word cloud");
    wordFrame.setLayout(new FlowLayout(FlowLayout.LEFT));

    for ( int i = 0; i < uniqueWords.size(); i++ ){
        int fontScale =
            2*((Integer)counts.get(i)).intValue();
    }

Watch out! These parens are necessary (so that the object is cast first, and then the method intValue is called).

NOTE: A better way to use ArrayLists to avoid casting (and the warnings the compiler will give you) is to use generics. See the section on Generics in the book (and we’ll get to these later this term).
Abstract class?
Abstract class example...

```java
public class GraphicLetter extends JPanel {
    public char whichChar() {
        return '\u0000';
    }

    public GraphicLetter makeCopy() {
        return null;
    }
}
```

These methods are “dummy” methods

Abstract class? Find out more in the reading!

```java
public abstract class GraphicLetter extends JPanel {
    public abstract char whichChar();

    public abstract GraphicLetter makeCopy();
}
```

In an abstract class we need not even define “dummy” methods.
import java.util.*;
public class Initials {
    private char[] firstChars; // FIELD

    public Initials (){       // CONSTRUCTOR
        // (1) get the number of characters to save (this makes sure it's a positive number)
        Scanner input = new Scanner(System.in);
        int numOfChars = -1;
        while (numOfChars <0){
            System.out.println("How many elements do you want in your array?");
            numOfChars = input.nextInt();
        }
        // IMPLEMENT HERE (2) and (3), to do as described below.
        [SOLUTION UPLOADED TO THE CLASS WEBSITE]
    } // END OF THE CONSTRUCTOR Initials()

    public String toString(){ return String.valueOf(this.firstChars);  }
}

} //END OF THE CLASS

Finish writing the constructor for this class Initials(): it takes no parameters and (1) prompts the user how many characters they would like to save in the array; (2) initializes the array firstChars: prompts the user to enter as many words as necessary, we will store the first char of each word in the array until it has been filled. (3) At the end of your method, the instance variable firstChars should refer to an array with the characters obtained from the user in sorted order. NOTE that step (1) is already done in the code!

Here is an example of how this constructor will work. The text in bold is typed by the user:
> Initials myInit = new Initials();
How many elements do you want in your array?
3
Enter 3 words:
 salut! Hola
hi
> System.out.println(myInit);
"Hhs"
public class Nim {

    // OBJECT SCOPE!!!
    public String[] players;
    public int[] board;

    public String toString() { return players[0] + " vs. " + players[1]; }

    public Nim(String name1, String name2, int[] theBoard) // CONSTRUCTOR SCOPE!!!
    {
        players = new String[2];
        players[0] = name1.toUpperCase();
        players[1] = name2;
        board = theBoard;
        theBoard = new int[board.length + 1];
    }

    public static void main(String[] args) // MAIN METHOD SCOPE
    {
        int[] board = {1, 5, 10, 15};
        String name1 = "Bear";
        String name2 = "Triton";
        Nim n = new Nim(name1, name2, board);
        board[3] = 10;
        name2 = name2.toLowerCase();
        System.out.println(name1); // 1ST ➔ Bear
        System.out.println(n);
        System.out.println(n.board[3]); // 3RD ➔ 10
    }
}

Midterm-2, p2

Follow the arrows from the right box (check the colors) and you’ll find the answer (DIAGRAM IN NEXT SLIDE)
class Person
{
    protected String name; // data member - protected

    public Person( String name ) { this.name = name; }
    public boolean isAsleep( int hr ) { return 22 < hr || 7 > hr; }
    public String toString() { return name; }

    public void status( int hr )
    {
        if ( this.isAsleep( hr ) )
            System.out.println( "Now offline: " + this );
        else
            System.out.println( "Now online: " + this );
    }
}

class Student extends Person
{
    protected int units; // additional data member

    public Student( String name, int units ) {
        super(name);
        this.units = units;
    }

    public boolean isAsleep( int hr ) // override
    { return 2 < hr && 8 > hr; }

    public String toString()
    {
        String result = super.toString();
        return result + " units: " + units;
    }
}
class CSEMajor extends Student
{
    protected boolean isTutor;

    public boolean isAsleep( int hr ) { return false; }

    public CSEMajor( String name, int units, boolean isTutor )
    {

    }

    public String toString()
    {

    }

    public static void main(String[] args)
    {
        Student s = new Student( "Sally", 16 );
        s.status( 7 ); // status at 7 am

        Person p = new CSEMajor( "Susan", 18, true );
        p.status( 3 ); // status at 3 am

        Student s = p;
        s.status( 3 ); // status at 3 am
    }
}

Try it at home!

What will these three status calls print?

A CSEMajor should print out as

Sally units: 42  Tutor? true

Do you think Java will be OK with this code?
More exercises!

class Person
{
    public String getName() {...}
    // no getMajor method defined here
}

class Student extends Person
{
    // no getName defined here
    public String getMajor() {...}
}

class General extends Person
{
    // no getName defined here
    public String getMajor() {...}
}

Person P  = new Person();
Person Q  = new Person();
Student S  = new Student();
General G = new General();
Object Ob = new General();

print( S.getName() );
P = S;
print( P.getMajor() );
G = Q;
G = Ob;

static void print(String s)
{
    System.out.println(s);
}

Which of the five lines of code below will work as it is written and which will Java complain about (and why?)

Is any casting needed?

If you are bored... write a method that takes a sorted ArrayList of ints and returns the int that occurs most frequently in the array.

// The ArrayList nums contains the elements 1, 2, 2, 2, 2, 3, 3
most( nums ) \rightarrow 2
Towards the Connect 4 GUI... DEMO!

NEXT WEEK! PSA6
Released on Thursday
We’ll talk about it on Tuesday