CSE 8B Today

PSA 5 due today

PSA6: Connect 4 returns
- Please start THIS WEEK (due a week from Thursday)
- I will poll you on Friday
Goal: EVERYONE will have “started”
- Lecture examples will tie in closely starting on Wednesday
- Speaking of your other feedback...

Exam 2 average: 68% → 84% overall
Nice job!!

ArrayLists (the array that never ends)

Exam 2 makeup as promised...
Feedback on Feedback (Keep)

• Worksheets and clicker questions
• Group discussions
• Running/tracing code + memory models
• Before/after slides
• PSAs:
  – Content
  – Examples in class
  – Partner work
Feedback on Feedback (Quit)

- Testing PSAs on new tests
  - I do this so that you will go beyond what we provide to test your own code
- Long discussion periods on clicker questions (but others say timing is about right)
  - Will try to adjust timing
- PSAs
  - Entirely: Sorry, not going to happen
  - Too hard: Will provide more support in class/discussion, and slightly longer time for harder ones.
  - Due day: It’ll float around a bit, but mostly Thursday. Sorry.
- Too many exams
  - I do this so that you can (a) get rapid feedback on your progress and (b) drop one.
Feedback on Feedback (Start)

• Reviews before tests
  – Discussion section Wednesday and, especially, Friday
  – I/TAs will post a few practice problems by Friday
• Tie lectures to the PSAs/more hints on PSAs
  – I will to a degree, but... this is what discussions are for
• Post “after” slides faster (and with correct answers)/write more on slides
• Cover topics for PSAs earlier:
  – Topics for PSAs are finished by the MONDAY of the week they are due. The previous Wednesday and Friday
• Reviewing PSAs after they are over (go to Friday’s discussion)
• More live coding
• More real-world examples
• More office hours
  – Please email me to find a time to meet!
• Cupcakes/pizza/pie?! (
If you said you want more info on the PSAs/review for the exam...

- Do you go to discussion section regularly?
  A. Yes: Mondays
  B. Yes: Wednesday
  C. Yes: Friday
  D. No
  E. I didn’t say I wanted more information
The time for individual clicking is...

A. Usually too short
B. Usually about right
C. Usually too long
The time for group discussion is ...

A. Usually too short
B. Usually about right
C. Usually too long
How many hours did you spend on PSA5 (Graphic Letter)?

A. Less than 3
B. 3-6
C. 6-9
D. More than 9
How many hours did you spend on PSA4 (Connect 4 board)?

A.  < 3
B.  3-6
C.  6-9
D.  More than 9

Tips for faster completion:
•  Read the PSA immediately
•  Don’t start coding until you understand exactly what you’re doing (seek help first!)
•  Work in the lab (again, starting early)
•  Test EVERY method before you move on
•  Methodical debugging with print statements
1. Name one thing that makes an abstract class different from a non-abstract class.

A. Instances of abstract classes can be created with the “new” operator, whereas instances of non-abstract classes cannot.

B. Regular classes have all of its methods defined with implementation, whereas abstract classes can have methods defined without implementation.

C. Regular classes can define methods without implementation, whereas abstract classes must define all of their methods with implementation.
2. Since an abstract method doesn’t have any implementation in an abstract class, where is it actually implemented?

A. In the superclass.

B. In another method in the abstract class.

C. In a subclass.

D. In another completely unrelated abstract class.
3. The following object is instantiated (like in the book):
GeometricObject geoObject = new Circle(10);
The method, getArea(), is an abstract method in the GeometricObject class.

When you call geoObject’s getArea() method, which class’s getArea() method is actually being called?

A. Circle
B. GeometricObject
C. Rectangle
D. Shape
class NewPanel extends JPanel {
    protected void paintComponent( Graphics g ) {
        super.paintComponent(g);
        g.drawLine( 0, 0, 50, 50 );
        g.drawString( "Banner", 0, 40 );
    }
}

The Graphics class in Java is an abstract class. So how is it possible that we can have an object that g references of type Graphics?

A. The object referenced by g is actually an instance of a (non-abstract) subclass of Graphics
B. It is not possible. This code will cause an error because you cannot instantiate a Graphics object
C. Graphics is not actually an abstract class in Java.
Another Use for Casting: ArrayLists

```java
public void setUniqueAndCounts(String[] words)
{
    int index = 0;
    int uniqueIndex = 0;
    int numUniqueWords = countUnique(words);
    String[] uniqueWords = new String[numUniqueWords];
    int[] 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
Another Use for Casting: ArrayLists

```java
class MyClass {
    public ArrayList counts;
    public ArrayList uniqueWords;

    public void setUniqueAndCounts( String[] words ) {
        counts = new ArrayList();
        uniqueWords = new ArrayList();
        int index = 0;
        int uniqueIndex = 0;
        while ( index < words.length ) {
            int count = getCount( words, index );
            uniqueWords.add(words[index]);
            counts.add(count);
            index += count;
        }
    }
}
```
Another Use for Casting: **ArrayLists**

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

public void setUniqueAndCounts( String[] words )
{
    counts = new ArrayList();
    uniqueWords = 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;
    }
}
```

Wrapper class
(An object that holds a primitive value)
Java adds this automatically (autoboxing)
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 an Integer, which when multiplied by 2 gives another Integer, and not an int
D. counts.get(i) will return an Object, which cannot be multiplied by 2
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*(((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).
Exam 3: Next Monday (!!)

• Review in this Friday’s discussion section (and some on Wednesday as well)
• Topics/skills:
  – Write a class that extends another class
  – Write a method that overrides a method in a base class
  – Reason about what will happen when calling methods in subclass/super-class objects
  – Describe how Java uses the type of the object at run-time to decide which method to call (Polymorphism)
  – Know when a reference to an object is legal, and when it will cause a compile error (e.g. a Person reference pointing to a Student object vs. a Student reference pointing to a Person object)
  – Identify and describe situations where casting is needed (and safe, or not safe)
  – Use the Graphics object to draw simple scenes
  – Describe how the paintComponent method is used to provide custom graphics in GUIs
Inheritance and Polymorphism summary

• At compile time (references matter):
  
  – The object must ALWAYS be the type of the reference that refers to it. If the reference type is Blah, then the object must ALWAYS be a Blah, including a subclass of Blah. For example, a Student is ALWAYS a Person, but a Person is not always a Student.
  
  – When an object is referenced by a variable, the reference type determines what the compiler thinks the object type is. E.g., you cannot reference a student’s units through a Person reference, even if the underlying object is a Student. Similarly, the following will cause a compile error:
    
    Person p = new Student( "Sally", 16 );  // OK!
    Student s = p;  // Error here: the compiler uses p to determine type of
    // RHS (NOTE ERROR ON PRINTOUT) object
  
  – Casting allows you to change the type of a reference at compile time. It will cause a runtime error if the actual object is not the type you are trying to cast to

• At run time (objects matter):
  
  – Java uses the actual type of the object to determine which methods/variables to use. The type of the reference no longer matters at runtime.
  
  – If you cast an object to a type that it is not, you will get a ClassCastException
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 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!

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

Is any casting needed?

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

1. `print( S.getName() );`
2. `P = S;`
3. `print( P.getMajor() );`
4. `G = Q;`
5. `G = Ob;`

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