CSE 8B Today

How was the exam?
A. No sweat!  B. Not too bad  C. So-so  D. Not so good  E. Totally awful!

Inheritance!

Person → Student → CSEMajor

JPanel → GraphicLetter → YOUR Graphic Letter

We’ll try to get the full alphabet so everyone will use it in PSA7

OUR HOME LAB is B230

PSA5! Out today
CSE 8B Today

Inheritance!

Person

Student

CSEMajor

Inheritance!

JPanel → GraphicLetter → YOUR Graphic Letter

Diagram showing relationships between classes and objects in Java and Swing packages.
Variable types vs. object types

Will the line above cause an error?
A. Yes – AT COMPILATION TIME!

The check to verify that variable type and object type (appear to) match is done at COMPILE TIME... but the ACTUAL TYPE of the object is used to determine which method to call at RUN TIME.
The "empty" constructor of the superclass will automatically be called by the subclass's constructor. If the superclass does not have a default constructor, you must explicitly invoke the super-class's constructor with the right parameters.
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 Foo, then the object must ALWAYS be a Foo (including any subclass of Foo). 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 field through a Person reference, even if the underlying object is a Student. Similarly, the following will cause a compile error:
    ```java
    Person p = new Student( "Sally", 16 ); // OK!
    Student s = p; // Error here: the compiler uses p to determine type of
                    // RHS 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
Inheritance Tree

Object

Person

Circus Performer

Teacher

Student

HistoryMajor

BiologyMajor

CSEMajor

Things to keep in mind!

a HistoryMajor is a Person…
"everything" is an Object…
PSA5 Inheritance Tree

- JComponent
  - JPanel
  - GraphicLetter
    - YOUR Graphic Letter (1)
    - YOUR Graphic Letter (2)

Things to keep in mind!
Your Graphic letter is a JPanel…
Why *whichChar* and *makeCopy*?

The LetterFactory (Holds GraphicLetters)

```
GraphicLetter_cs8szz1
GraphicLetter_cs8szy1
GraphicLetter_cs8sdr1
GraphicLetter_cs8suv2
GraphicLetter_cs8sim2
```

“Give me a Z!”
(The LetterFactory knows the objects are GraphicLetters, but doesn’t know which ones)
Why whichChar and makeCopy?

The LetterFactory

GraphicLetter_cs8szz1
GraphicLetter_cs8szy1
GraphicLetter_cs8sdr1
GraphicLetter_cs8suv2
GraphicLetter_cs8sim2

whichChar are you?

‘A’

“Give me a Z!”
Why whichChar and makeCopy?

JComponent

JPanel

GraphicLetter

YOUR Graphic Letter (1)

YOUR Graphic Letter (2)

The LetterFactory

GraphicLetter_cs8szz1
GraphicLetter_cs8szy1
GraphicLetter_cs8sdr1
GraphicLetter_cs8suv2
GraphicLetter_cs8sim2

whichChar are you?

‘D’

“Give me a Z!”
Why whichChar and makeCopy?

JComponent

JPanel

GraphicLetter

YOUR Graphic Letter (1)

YOUR Graphic Letter (2)

The LetterFactory

GraphicLetter_cs8szz1

GraphicLetter_cs8szy1

GraphicLetter_cs8sdr1

GraphicLetter_cs8suv2

GraphicLetter_cs8sim2

whichChar are you?

‘Z’

Great! makeCopy of yourself!

“Give me a Z!”
In what class is the `drawLine` method defined?

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

Who calls the paintComponent method in NewPanel?
A. Java automatically calls this method when it paints (or repaints) the component
B. The programmer must call this method directly on the NewPanel object
C. Java will automatically call it, but the programmer may also call it if she or he wants the painting to happen immediately.

http://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html