CSE 8B Today

Inheritance!

Person

Student

CSEMajor

JPanel → GraphicLetter → YOUR Graphic Letter

Classes in the java.awt package

Panel → Applet → JApplet

Window → Frame → JFrame

Dialog

Swing GUI components such as JButton, JLabel, JTextField, JPanel, etc.

Lightweight

Swing Components in the javax.swing package
Variable types vs. object types

Student s = new Person(“Sally”);

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
a HistoryMajor is a Person…
"everything" is an Object…
Things to keep in mind!
Your Graphic letter *is a JPanel*…

Things to keep in mind!

---

**JComponent**

**JPanel**

**GraphicLetter**

**YOUR Graphic Letter (1)**

**YOUR Graphic Letter (2)**

**GraphicLetter_cs8szz1**
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?

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?
‘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_cs8suv2

GraphicLetter_cs8sim2

whichChar are you?

‘Z’

Great! makeCopy of yourself!

“Give me a Z!”
class NewPanel extends JPanel {
    protected void paintComponent( Graphics g )
    {
        super.paintComponent(g);
        g.drawLine( 0, 0, 50, 50 );
        g.drawString( "Banner", 0, 40 );
    }
}

In what class is the drawLine method defined?
A. NewPanel
B. JPanel
C. Graphics
D. paintComponent
E. Other
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.
Polymorphism

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)
public void foo(){
    x = 5;
    while ( x != 8 ) {
        x += 2;
        S.o.p( “Spam!” );
        break;
    }
    S.o.p( “Spam!” );
}

How many times does this code print Spam!?
A. 1
B. 2
C. 3
D. 4
E. Infinite

public void foo(){
    x = 5;
    while ( x != 8 ) {
        x += 2;
        S.o.p( “Spam!” );
        if (x == 7){
            break;
        }
    }
    S.o.p( “Spam!” );
}

How many times does this code print Spam!?
A. 1
B. 2
C. 3
D. 4
E. Infinite