(Slides continue from last time)

How was the exam?

• A. No problem
• B. Not too bad
• C. So-so
• D. Pretty hard
• E. Totally brutal

(not counting #3)
```java
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 void study() { // override
        System.out.println("I'm studying");
    }
    public boolean isAsleep(int hr) { // override
        return 2 < hr && 8 > hr;
    }

    public String toString() {
        String result = super.toString();
        return result + " units: " + units;
    }
}
```

**Polymorphism**

```java
Polymorphism
```

```
All students are people!

"We always start looking in the class for the object that we have"

What will this code print?
A. Now online: Sally units: 16
B. Now offline: Sally
C. This code has a compile error
D. This code has a run-time error
E. I don't know

in main:
```
Person p;
p = new Student("Sally", 16);
p.status(1);
```

"Sally"

```
P 16
```

in main:
```
Person p; p = new Student("Sally", 16);
p.status(1);
p.study(); // compile error
```

"Sally"
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 Person p;
p = new Student( "Sally", 16 );
p.status( 1 );
Polymorphism

Sometimes the exact type is known at compile-time:

"Ordinary" code -- exact type known at compile time:

```java
Student s = new Student( "Sally", 16 );
then we can use s.units or s.isAsleep ...
```
But sometimes it's 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.

This is legal too (and may be very useful!):

```java
Person p;
later on…
p = new Student( "Sally", 16 );
```
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;
    }
}

In main:
Student s;
s = new Person( "Sally" );
s.status( 1 );

What will this code print?
A. Now online: Sally units: 0
B. Now offline: Sally
C. This code has a compile error
D. This code has a run-time error
E. I don’t know
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;
    }
}

Inheritance Details

All People are NOT Students. If you store a Person in a variable that is supposed
to reference a Student, the compiler gets nervous (and gives you an error)
because you might try to make that object do something that it cannot do.

This is true EVEN IF you never actually ask it to do something it cannot do (as in
this example).

in main:
Student s;
s = new Person( "Sally" );
s.status( 1 );
Inheritance Tree

Object

Person

Circus Performer

Teacher

Student

HistoryMajor

BiologyMajor

CSEMajor

"everything" is an Object…
a HistoryMajor is a Person…

Things to keep in mind!
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…
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.