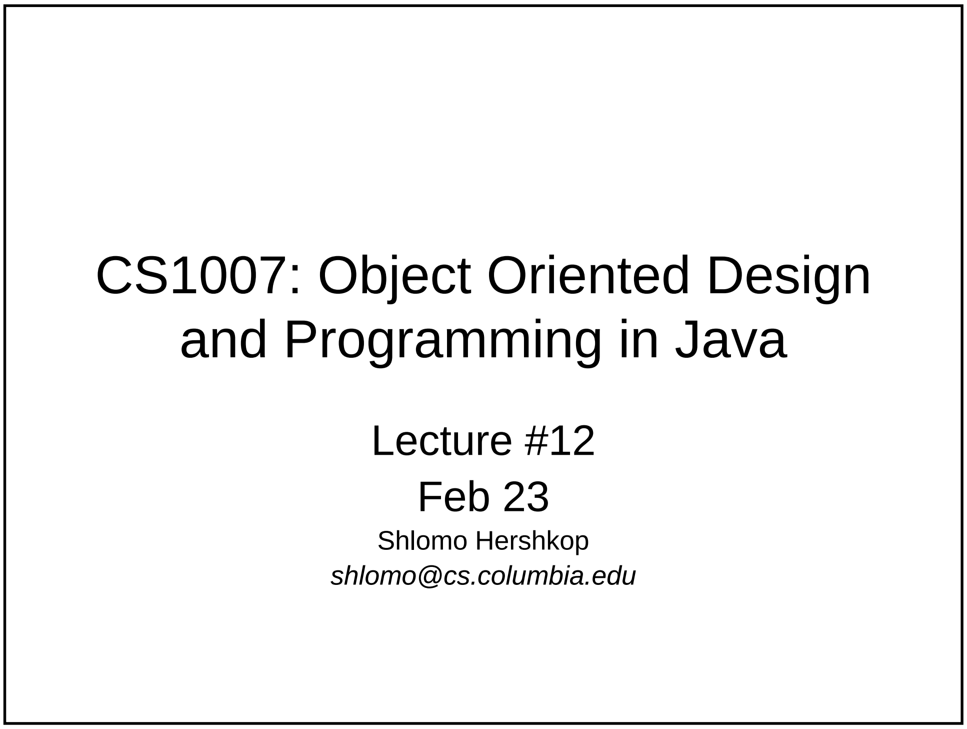
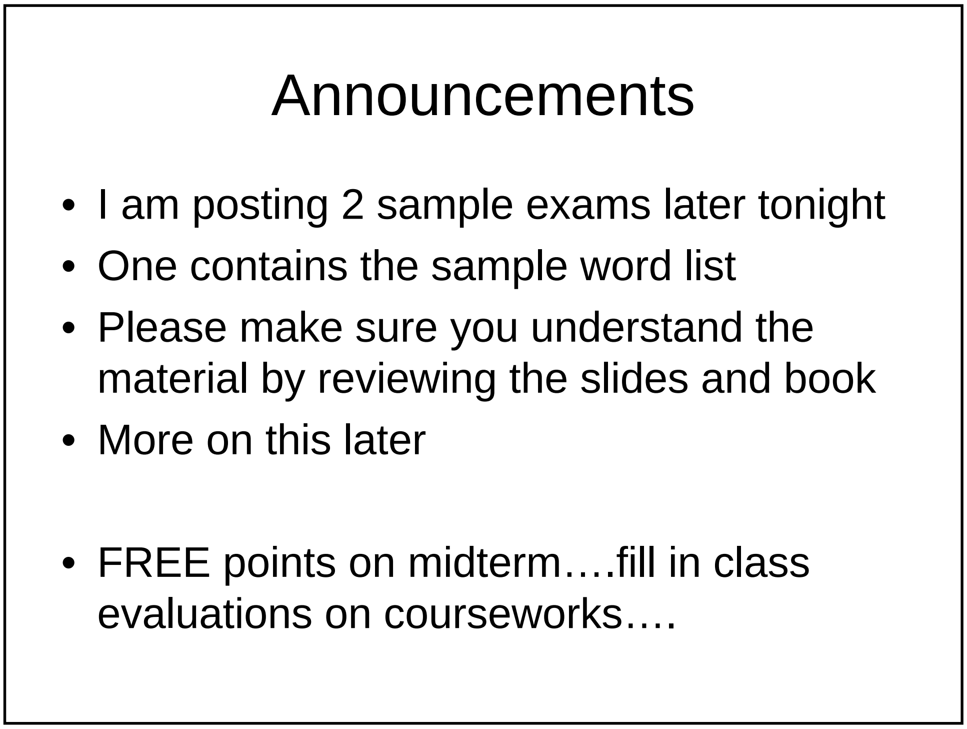


# CS1007: Object Oriented Design and Programming in Java

Lecture #12

Feb 23

Shlomo Hershkop
*shlomo@cs.columbia.edu*



## Announcements

- I am posting 2 sample exams later tonight
- One contains the sample word list
- Please make sure you understand the material by reviewing the slides and book
- More on this later

- FREE points on midterm….fill in class evaluations on courseworks….

## Outline

- Wrap up of random topics
  - Java classes
  - Graphics low level
  - Inheritance (with code)
- Will review for midterm
- If I leave out anything please come to office hours

## Overloading

- What was that again?
- example

# Overloading

- Can be in the same class

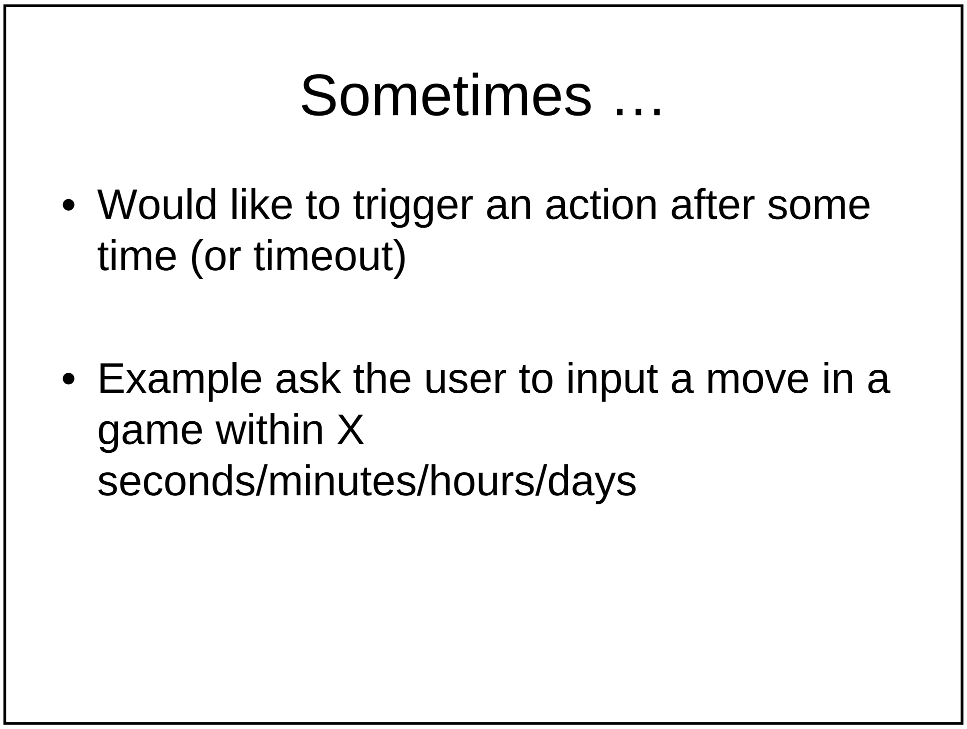
- Can be in subclass

- Late binding

# GUI programming

- Remember that GUI based programming
  - Usually will setup the gui
  - Define actionlisteners
  - Show up on screen

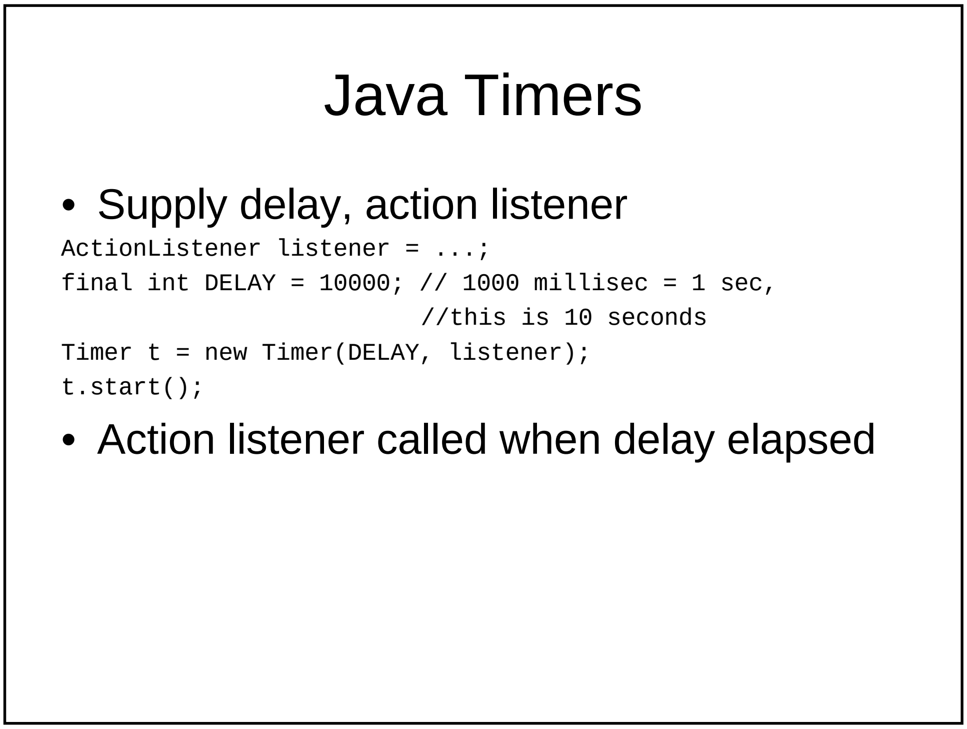
  - Wait……for something



# Sometimes …

- Would like to trigger an action after some time (or timeout)

- Example ask the user to input a move in a game within X seconds/minutes/hours/days



# Java Timers

- Supply delay, action listener

```
ActionListener listener = ...;
final int DELAY = 10000; // 1000 millisec = 1 sec,
                         //this is 10 seconds
Timer t = new Timer(DELAY, listener);
t.start();
```

- Action listener called when delay elapsed

# Example update a clock

```
      ActionListener listener = new
         ActionListener()
         {
            public void actionPerformed(ActionEvent event)
            {
               Date now = new Date();
               textField.setText(now.toString());
            }
         };
      final int DELAY = 1000;
         // Milliseconds between timer ticks
      Timer t = new Timer(DELAY, listener);
      t.start();
```

# Be careful

- java.util.Timer
- java.swing.Timer

- How does java figure out which class Timer you are referring to??

# Low Level Java Drawing

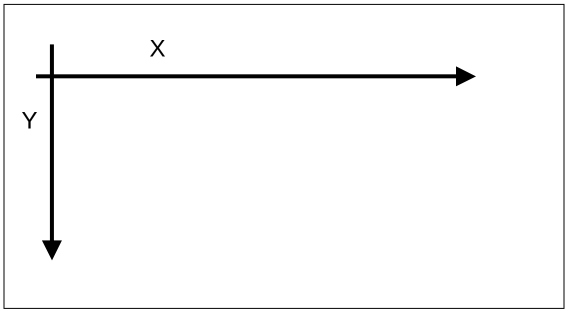
- All drawing done on graphic context objects
- Canvas is the area to draw on
- Use a “Graphics” or “Graphics2D” objects to interact with canvas
- It is instantiated to the paint() or update() method

# Triggers

- Paint is the method to actually do the paint on the screen
- Will be triggered
  - When cal setVisible
  - When call paint/repaint
  - When maximize after minimize
  - When move screen (sometimes)

# Remember Coordinate System

- `(0,0)` on top left corner
- X increase to right
- Y increase downwards
- `(0,object.getwidth() )` is upper right corner



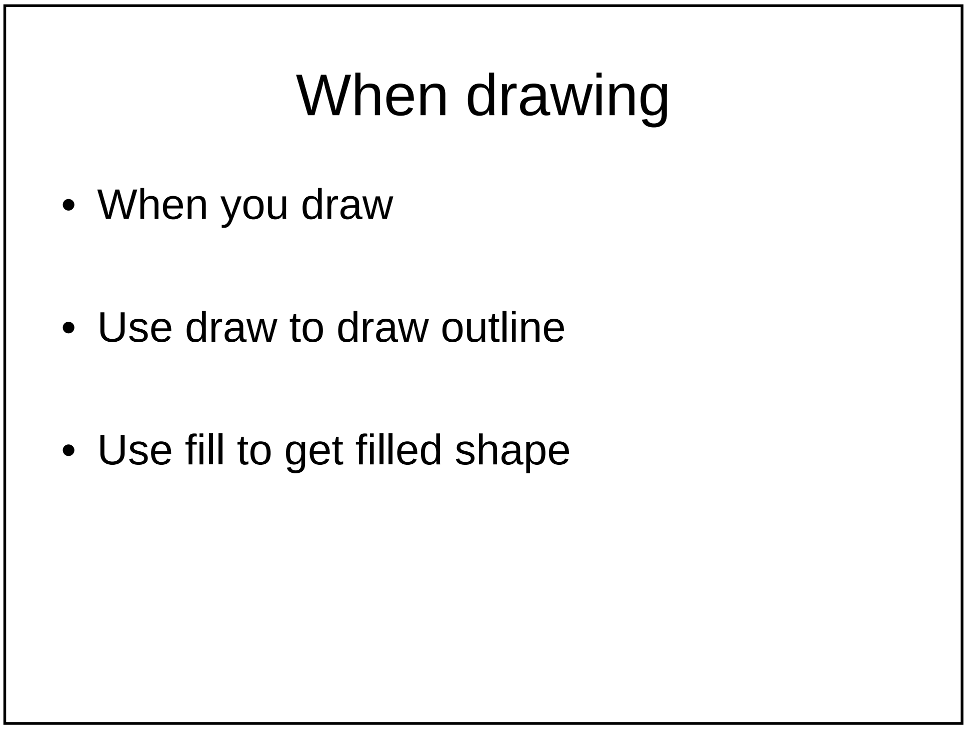
# Drawing Shapes

- paintIcon method receives graphics context of type Graphics
- Actually a Graphics2D object in modern Java versions

```
public void paintIcon(Component c, Graphics g, int
  x, int y)
{
 Graphics2D g2 = (Graphics2D)g;
 . . .
}
```

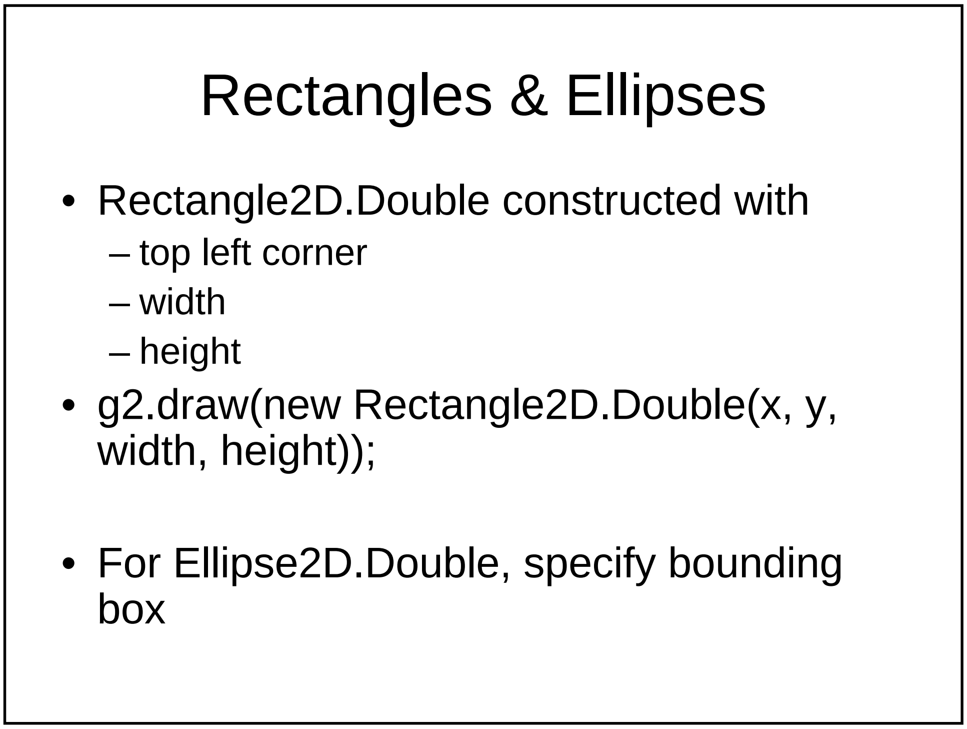
- Can draw any object that implements Shape interface

```
Shape s = . . .;
g2.draw(s);
```



# When drawing

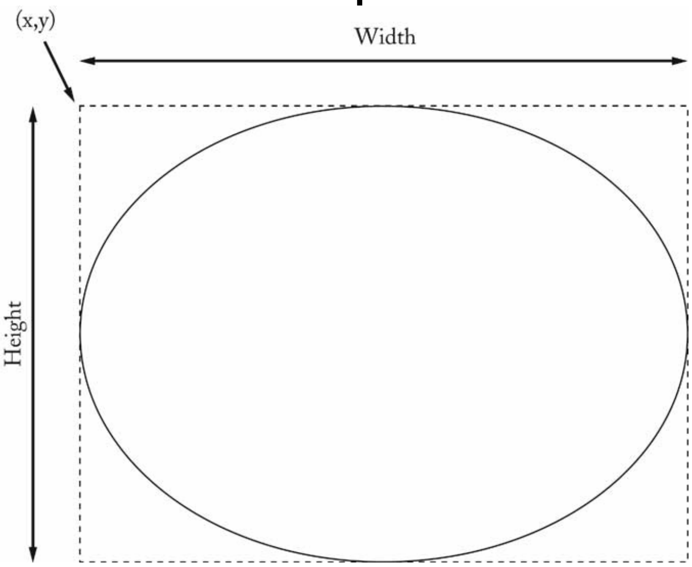
- When you draw
- Use draw to draw outline
- Use fill to get filled shape



# Rectangles & Ellipses

- Rectangle2D.Double constructed with
  - top left corner
  - width
  - height
- g2.draw(new Rectangle2D.Double(x, y, width, height));
- For Ellipse2D.Double, specify bounding box

# Ellipses



```
Shape rectangle = new
   rectangle2D.Double(x, y, width, height);

g2.draw(rectangle);
```

# Line Segments

- `Point2D.Double` is a point in the plane
- `Line2D.Double` joins to points

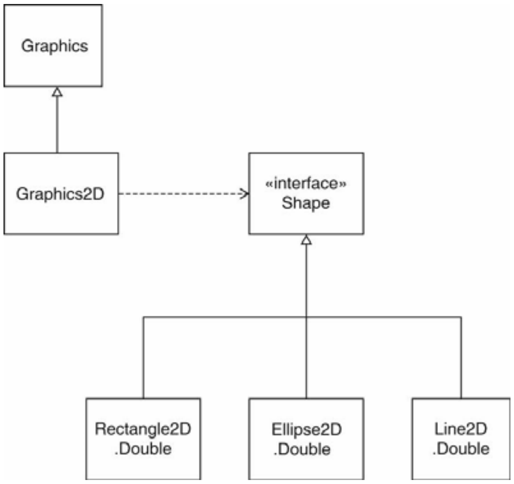
```
Point2D.Double start = new Point2D.Double(x1, y1);

Point2D.Double end = new Point2D.Double(x2, y2);

Shape segment = new Line2D.Double(start, end);

g2.draw(segment);
```

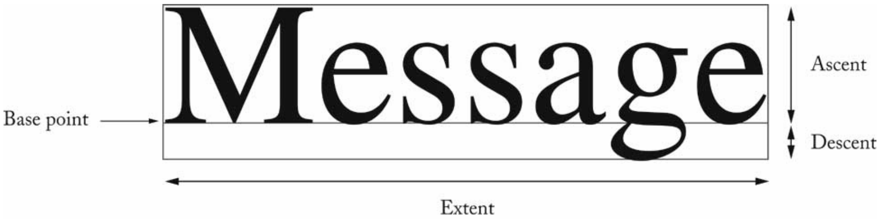
# Relationships between shape classes



- Code example
- Shapeexample.java
- Myrawgui.java

## Besides Shapes can Draw

- `g2.drawString(text, x, y);`
- `x, y` are base point coordinates



# Review for midterm

- I’d like to quickly address things which should put you at ease about the exam

# First

- I’m not out to get you
- It’s a test of what you understand…not memorized…hence its open book
- If you aren’t sure on how to create an exception….you need to see myself or the TA (or reread the class notes)

# UML

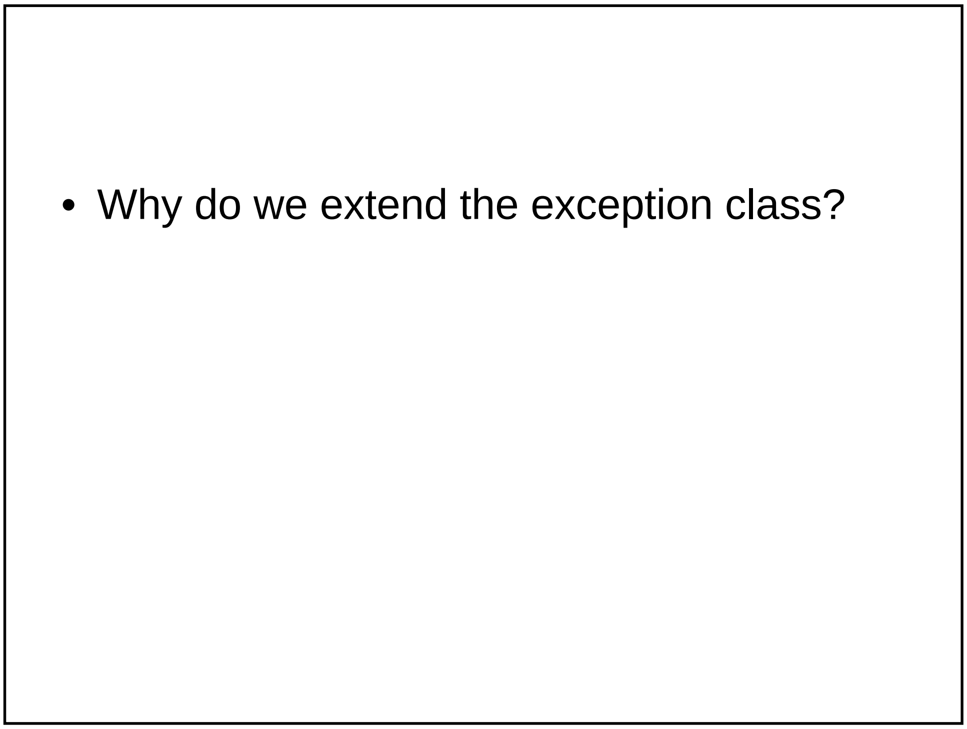
- Should understand the theory behind it and different types of documentation
- You are being tested on this in the hw assignments

- Unfortunately there isnt enough time on exams for good long examples.

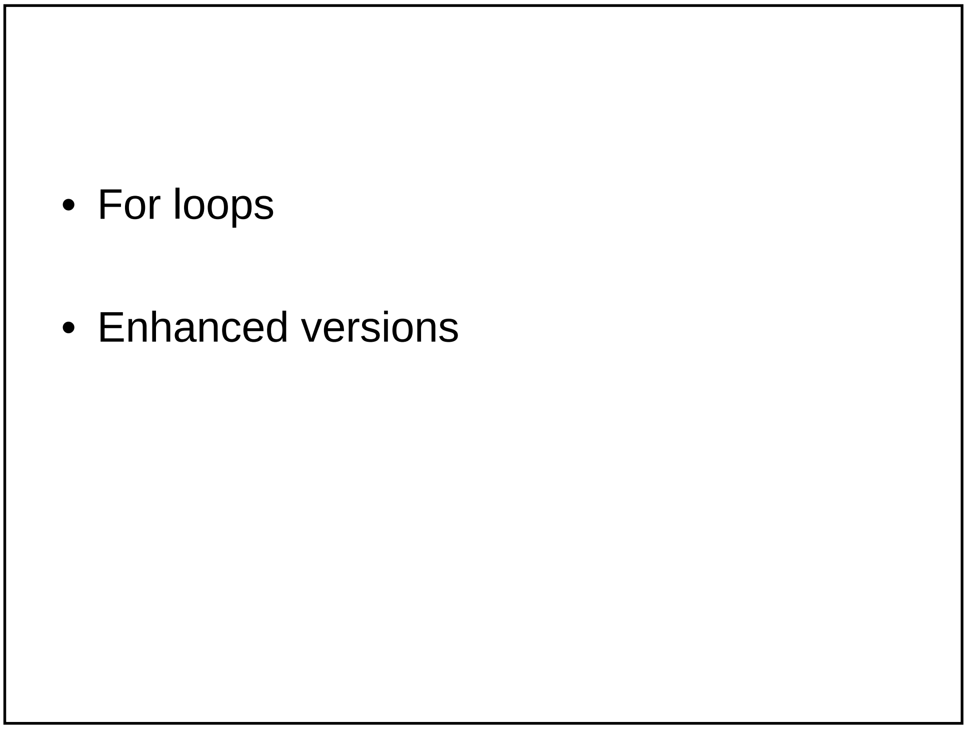
# Hard exception example

- Try
- Catch
- Finally

- Understand how they interact



- Why do we extend the exception class?



- For loops

- Enhanced versions

- What does window.pack() do?

- casting

- Understanding static methods and when to use them

- Understanding generics