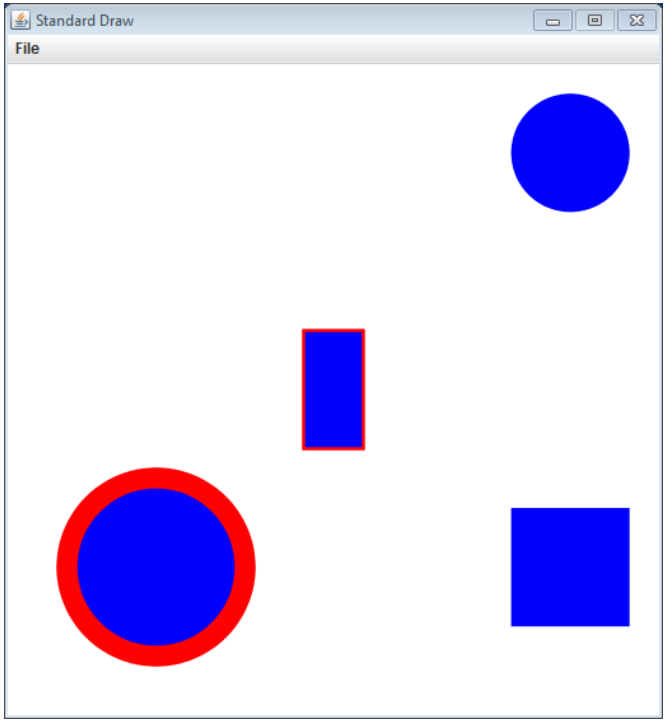
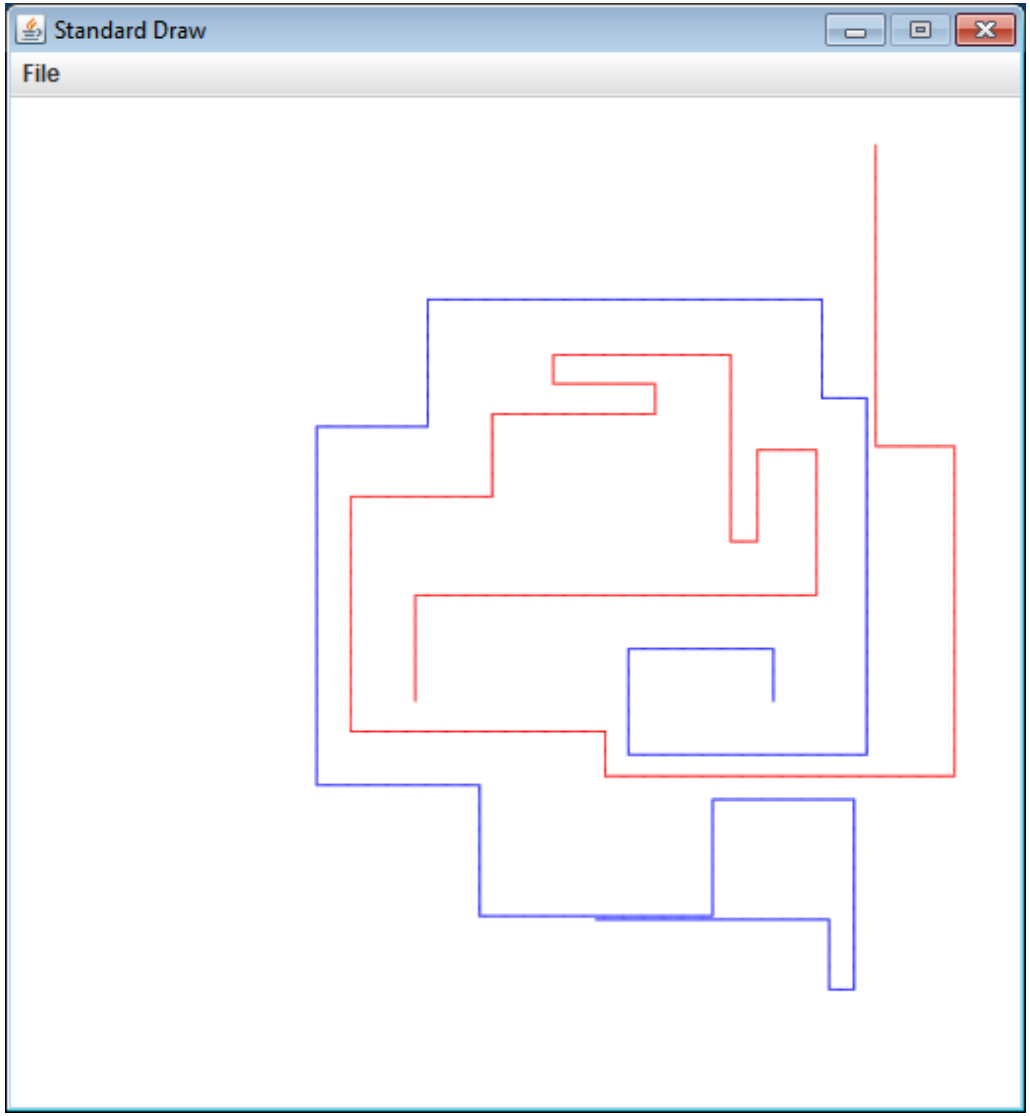
Interfaces and sorting



Overview

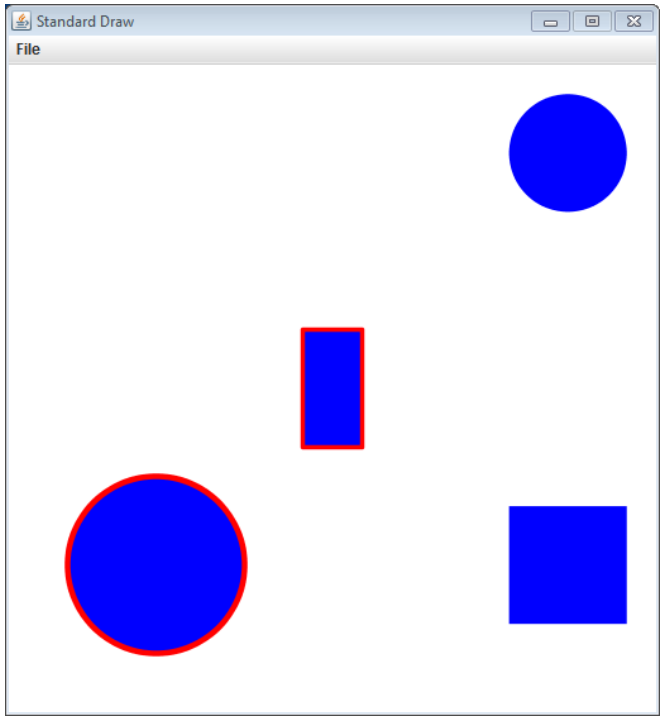
- Finish the snake game
  - Add a second player
  - Check if players hit a tail
- A shape object hierarchy
  - Classes that "extend"
  - Classes that "implement"
- Sorting objects
  - For simple classes
  - For your own classes

Snake game



Shape object hierarchy

- Represent shapes that can:
  - Draw themselves
  - Test for intersection with (x, y) coordinate
  - Change color
  - Support:
    - Circles
    - Rectangles
    - Circles with borders
    - Rectangles with borders



Shape

```
protected double x, y;
protected Color color;

public double getX();
public double getY();
public void setColor(double r, double g, double b);
public double distance(double x, double y);
public abstract void draw();
public abstract boolean intersects(double x, double y);
```

Circle extends Shape

```
protected double radius;

public void draw();
public boolean intersects(double x,
                          double y);
```

Rectangle extends Shape

```
protected double width, height;

public void draw();
public boolean intersects(double x,
                          double y);
```

CircleBorder extends Circle

```
protected double thickness;
protected Color borderColor;

public void draw();
public void setBorderColor(double r,
                           double g,
                           double b);
public void changeBorderThickness(double d);
public void setBorderThickness(double d);
```

RectangleBorder extends Rectangle

```
protected double thickness;
protected Color borderColor;

public void draw();
public void setBorderColor(double r,
                           double g,
                           double b);
public void changeBorderThickness(double d);
public void setBorderThickness(double d);
```

Shape

```
protected double x, y;
protected Color color;

public double getX();
public double getY();
public void setColor(double r, double g, double b);
public double distance(double x, double y);
public abstract void draw();
public abstract boolean intersects(double x, double y);
```

1) Can we create a Shape object?
2) Which are abstract classes?
3) Which are concrete classes?
4) Which are subclasses of Shape?
5) Which are subclasses of Circle?
6) Which methods are overridden?
7) Can we change radius to private?

Circle extends Shape

```
protected double radius;

public void draw();
public boolean intersects(double x,
                          double y);
```

Rectangle extends Shape

```
protected double width, height;

public void draw();
public boolean intersects(double x,
                          double y);
```

CircleBorder extends Circle

```
protected double thickness;
protected Color borderColor;

public void draw();
public void setBorderColor(double r,
                           double g,
                           double b);
public void changeBorderThickness(double d);
public void setBorderThickness(double d);
```

RectangleBorder extends Rectangle

```
protected double thickness;
protected Color borderColor;

public void draw();
public void setBorderColor(double r,
                           double g,
                           double b);
public void changeBorderThickness(double d);
public void setBorderThickness(double d);
```

Border objects

- CircleBorder and RectangleBorder
  - Share two identical instance variables
  - Share three identical methods
  - Can we somehow consolidate?
    - Not really since we want to be related to Shape
    - But shapes like Circle and Rectangle don't want borders

CircleBorder extends Circle

```
protected double thickness;
protected Color borderColor;

public void draw();
public void setBorderColor(double r,
                           double g,
                           double b);
public void changeBorderThickness(double d);
public void setBorderThickness(double d);
```

RectangleBorder extends Rectangle

```
protected double thickness;
protected Color borderColor;

public void draw();
public void setBorderColor(double r,
                           double g,
                           double b);
public void changeBorderThickness(double d);
public void setBorderThickness(double d);
```

Interfaces

- Java interfaces
  - Java's alternative to multiple inheritance
  - Classes promise to implement same API
    - An interface is just a list of abstract methods
    - If two classes implement same interface, they can live in the same array for polymorphic goodness
  - Example uses of interfaces:
    - Allow any object type to be easily sorted
    - GUI event listeners (e.g. when a button is pushed)
    - Classes that can run in their own thread

Bordered interface

```
// Interface for a shape that has a border that can be a
// different color and has a variable pen thickness.
public interface Bordered
{
   public abstract void setBorderColor(double r,
                                       double g,
                                       double b);

   public abstract void setBorderThickness(double thickness);

   public abstract void changeBorderThickness(double delta);
}
```

A class adds "implements Bordered" to the class declaration.

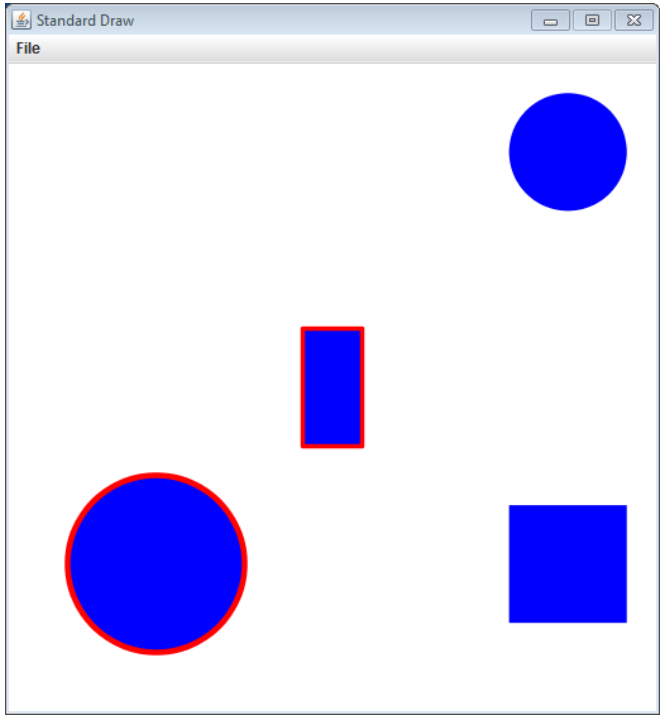
The class must then implement the three methods in interface Bordered.

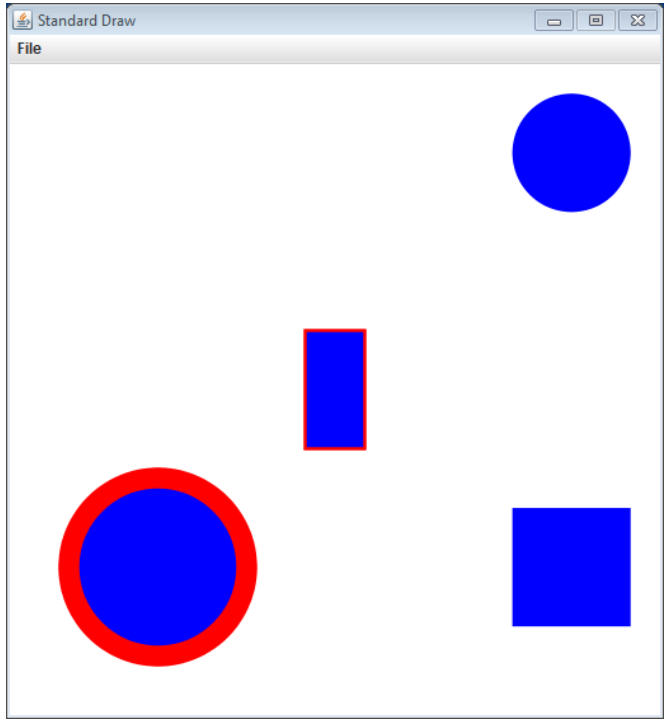
```
public class CircleBorder extends Circle implements Bordered
```

```
public class RectangleBorder extends Rectangle implements Bordered
```

GrowShape

- Show a bunch of Shape objects
  - If mouse is over a shape:
    - Temporarily change object's color
    - If object has a border, permanently grow the border





GrowShape

```
public static void main(String [] args)
{
    Shape [] shapes = new Shape[4];

    shapes[0] = new RectangleBorder(0.5, 0.5, 0.1, 0.2);
    shapes[1] = new CircleBorder(0.2, 0.2, 0.15);
    shapes[2] = new Circle(0.9, 0.9, 0.1);
    shapes[3] = new Rectangle(0.9, 0.2, 0.2, 0.2);

    while (true)
    {
       StdDraw.clear();
       double x = StdDraw.mouseX();
       double y = StdDraw.mouseY();
       for (Shape shape : shapes)
       {
          if (shape.intersects(x, y))
          {
             shape.setColor(0.3, 0.1, 0.5);
             if (shape instanceof Bordered)
                ((Bordered) shape).changeBorderThickness(0.001);
          }
          else
             shape.setColor(0.0, 0.0, 1.0);

          shape.draw();
       }
       StdDraw.show(100);
    }
}
```

Polymorphic array holding objects in the `Shape` hierarchy. Some have borders, some don't.

Only increase the border on objects that implement the `Bordered` interface.

You must check using `instanceof` before attempting to cast object.

Sorting

- Goal: Print out a String array alphabetically
  - Easy one-liner for built-in types

```
import java.util.Arrays;

public class SortString
{
   public static void main(String [] args)
   {
      String [] names = {"dave", "bob", "alice", "carol", "eve"};

      Arrays.sort(names);
      for (String name : names)
         System.out.println(name);
   }
}
```

```
% java SortString
alice
bob
carol
dave
eve
```

WordProb class

- WordProb: Stores a word and a probability

```
public class WordProb
{
    private String word = "";
    private double prob = 0.0;

    public WordProb(String word, double prob)
    {
       this.word = word;
       this.prob = prob;
    }

    public String toString()
    {
       return word + " " + prob;
    }
}
```

```
public class WordProbClient
{
   public static void main(String [] args)
   {
       WordProb [] items = new WordProb[4];

       items[0] = new WordProb("hello", 0.005);
       items[1] = new WordProb("the", 0.02);
       items[2] = new WordProb("zebra", 0.001);
       items[3] = new WordProb("if", 0.01);




   }
}
```

WordProb class

- Goal: Sort entries in probability order

```
public class WordProb
{
    private String word = "";
    private double prob = 0.0;

    public WordProb(String word, dou
    {
       this.word = word;
       this.prob = prob;
    }

    public String toString()
    {
       return word + " " + prob;
    }
}
```

```
% java WordProbClient
Exception in thread "main" java.lang.ClassCastException:
WordProb cannot be cast to java.lang.Comparable
          at java.util.Arrays.mergeSort(Arrays.java:1144)
          at java.util.Arrays.sort(Arrays.java:1079)
          at WordProbClient.main(WordProbClient.java:15)
```

```
import java.util.Arrays;

public class WordProbClient
{
   public static void main(String [] args)
   {
       WordProb [] items = new WordProb[4];

       items[0] = new WordProb("hello", 0.005);
       items[1] = new WordProb("the", 0.02);
       items[2] = new WordProb("zebra", 0.001);
       items[3] = new WordProb("if", 0.01);

       Arrays.sort(items);
       for (WordProb item : items)
          System.out.println(item);
   }
}
```

Comparable interface

```
public class WordProb implements Comparable
{
    private String word = "";
    private double prob = 0.0;

    public WordProb(String word, double prob)
    {
       this.word = word;
       this.prob = prob;
    }

    public String toString()
    {
       return word + " " + prob;
    }

    public int compareTo(Object o)
    {
       if ((o != null) && (o instanceof WordProb))
       {
          WordProb other = (WordProb) o;
          if (prob > other.prob)
             return -1;
          else if (prob < other.prob)
             return 1;
          else
             return 0;
       }
       return -1;  // default if o is not a WordProb
    }
}
```

We promise to implement a `compareTo()` method.

Required for the `Arrays.sort()` method to work.

Returns negative number if calling object "comes before" the parameter o.

Returns positive number if calling object "comes after" the parameter o.

Returns zero if "equal"

```
% java WordProbClient
zebra 0.0010
hello 0.0050
if 0.01
the 0.02
```

Summary

- Developed Shape object hierarchy
  - Circle and Rectangle branches
  - CircleBorder and RectangleBorder
    - Share functionality with each other
    - But not with their parent classes
- Java interfaces
  - 100% abstract class
    - A promise to implement a set of methods
    - Objects unrelated by inheritance can live in same array
    - A class can implement multiple interfaces
  - Used by Java for sorting objects and more