Hiding your variables

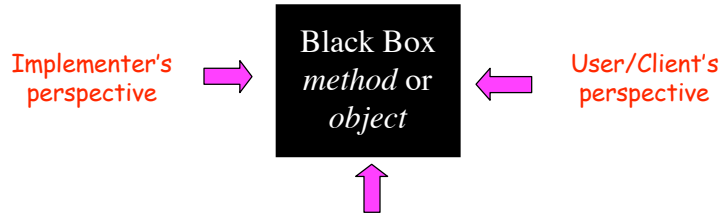
Data abstractions

Tuesday, November 21, 2006

CS111 Computer Programming

Department of Computer Science
Wellesley College

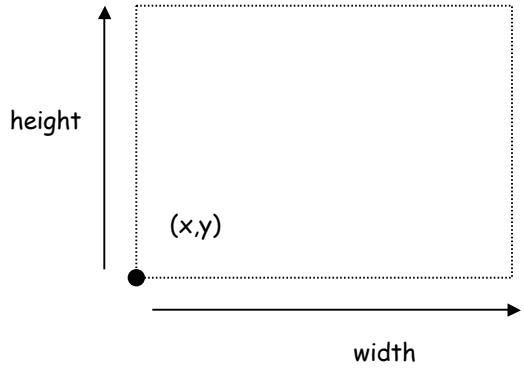
Data abstraction



Contract
describes rules for use
(acts as an **abstraction barrier** that separates
the interface from the implementation)

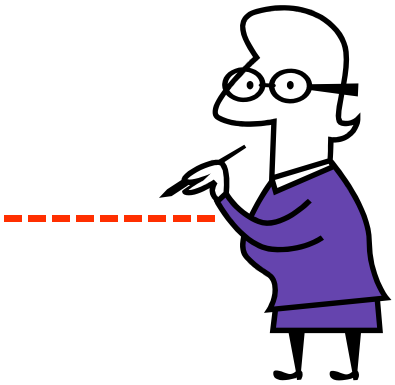
*The user (client) doesn't know or care how the black box works, she just wants to use it. This is DATA ABSTRACTION!!!

Our very own rectangle class



Negotiating a contract

- Who is going to use our rectangle class and why?
- What methods will our client require to get her job done.
- Some negotiation may be necessary before both sides are ready to sign on the dotted line.



The Rect Contract: Constructors

```
public Rect (int x, int y, int w, int h)
```
Constructs a Rect with lower left point (x, y), width w, and height h.

```
public Rect (Point p, int w, int h)
```
Constructs a Rect with lower left point p, width w, and height h. Subsequent changes to p should *not* affect the rectangle.

```
public Rect (Point p1, Point p2)
```
Constructs a Rect with lower left point p1 and upper right point p2. Subsequent changes to p1 and p2 should *not* affect the rectangle.

```
public Rect()
```
Constructs a Rect with lower left point (0,0), width 200, and height 100.

* We use the name Rect rather than Rectangle to avoid confusion with the java.awt.Rectangle class.

The Rect Contract: Instance Methods

```
public int getWidth()
```
Return the width of this rectangle

```
public int getHeight()
```
Return the height of this rectangle

```
public Point getLLpt()
```
Return the rectangle's lower left point. Subsequent changes to this point should *not* affect this rectangle.

```
public Point getURpt()
```
Return the rectangle's lower left point. Subsequent changes to this point should *not* affect this rectangle.

```
public int area()
```
Return the area of this rectangle.

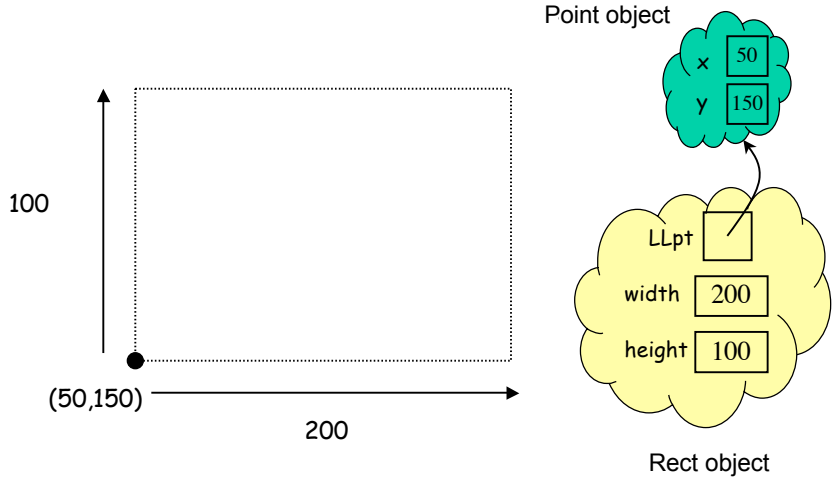
```
public String toString()
```
Return a string representation of this rectangle.

We could add many more instance methods …

Representing our rectangle

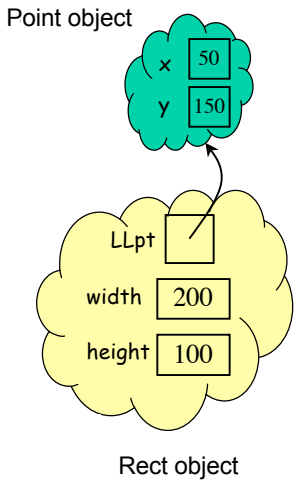
- How the state of an object is represented is ENTIRELY up to the programmer.
- The interface may or may not relate directly to the specific instance variables.

One possible representation



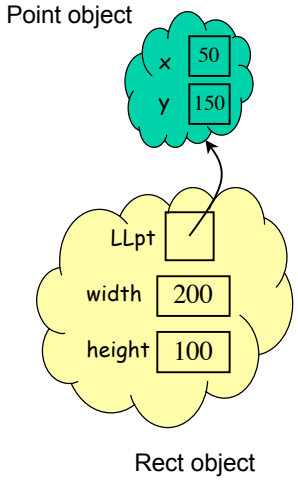
Implementing our contract

```
public class Rect
{

  // instance variables

  ...

  // constructors

  ...

  // instance methods

  ...

}
```



Instance variables

```
public class Rect {
     // instance variables
     private Point LLpt;
     private int width;
     private int height;

     // constructors

     ...

     // instance methods

     ...

}
```

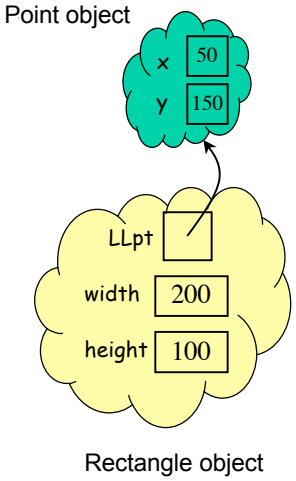


Constructors

```
public class Rect {
  // instance variables
  private Point LLpt;
  private int width;
  private int height;

  // constructors
  public Rect(int x, int y, int w, int h){

  }
  public Rect(Point p, int w, int h) {

  }
  public Rect(Point p1, Point p2) {

  }
  public Rect() {      …

  }

  // instance methods...
}
```

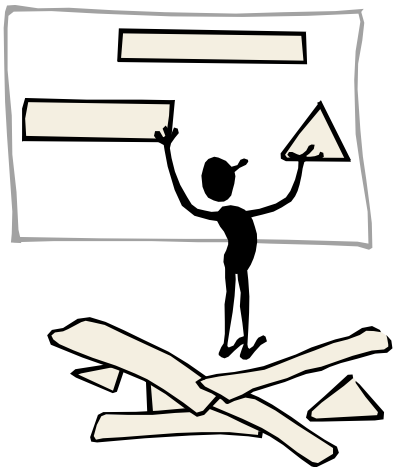

Instance methods

```
public class Rectangle {
  // instance variables and constructor methods …

  // instance methods
  public int getWidth() {

  }
  public int getHeight(){

  }
  public Point getLLpt() {

  }
  public int getURpt(){

  }
  public int area(){

  }
  public String toString(){

  }

}
```



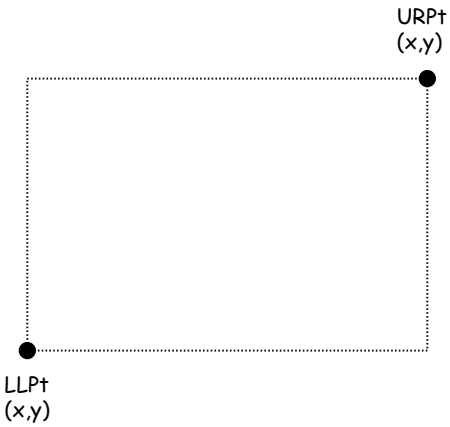
Changing our mind

- It is possible that someday we become dissatisfied with our design decisions and wish for a different implementation of Rectangles.
- As long as we fulfill the same contract*, no problem.

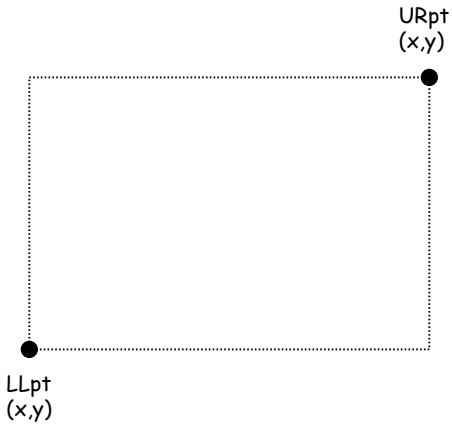


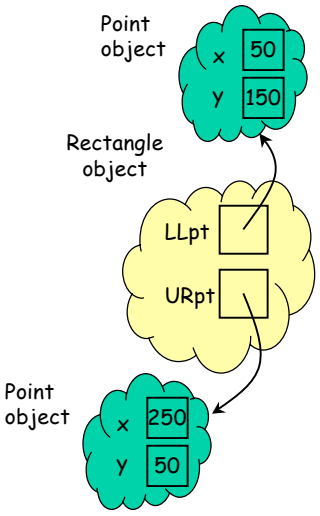
*And we have kept our data private.

A new representation



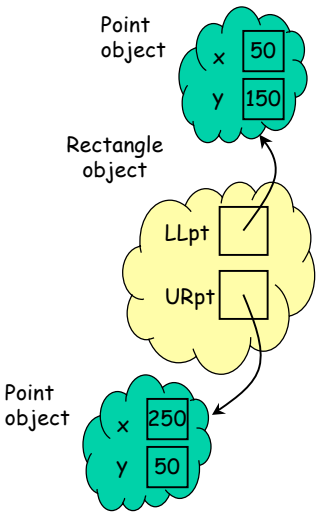
A new representation





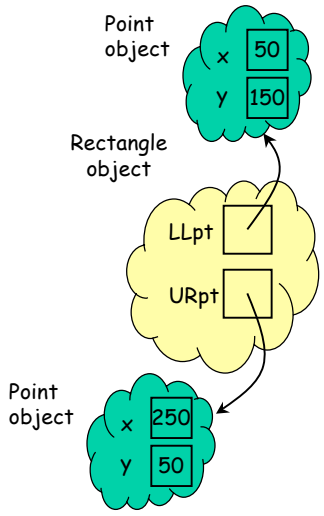
Here we go again

```
public class Rect
{
      // instance variables
      ...

      // constructors
      ...

      // instance methods
      ...

}
```



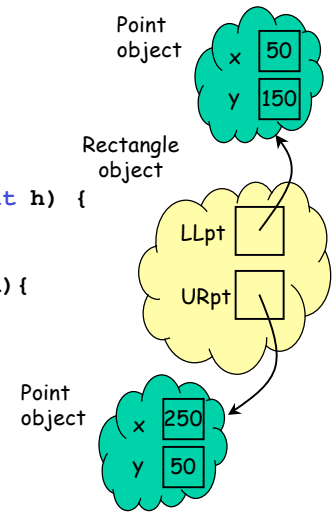
Thank goodness for private variables

```
public class Rect
{
     // instance variables
     private Point LLpt;
     private Point URpt;

     // constructors
     ...


     // instance methods
     ...


}
```



New constructors

```
public class Rect {
  // instance variables
  private Point LLpt;
  private Point URpt;

  // constructors
  public Rect(int x,int y, int w,int h) {

  }
  public Rect(Point p, int w, int h){

  }
  public Rect(Point p1, Point p2){

  }
  public Rect(){
  }
     // instance methods
}
```



Instance methods

```
public class Rect {
     // instance variables & constructors
     ...
     // instance methods
  public int getWidth(){

  }
  public int getHeight(){

  }
  public Point getLLpt(){

  }
  public int getURpt(){

  }
  public int area(){

  }
  public String toString(){

  }
}
```