


CS101

Problem Solving and Object-Oriented Programming

L15: Primitive Data Types and Signatures



Combining Multiple Conditionals

- `&&` (and) combines adjoining conditions in a way that the final result will be true only if all are true
 - Ex: `a && b && c`
is true if a,b,c are true
 - `||` (or) combines adjoining conditions in a way that if any of them is true, the final result will be true
 - Ex: `a || b || c`
is true if any of a, b, c, is true
- 



The Craps Example

- A block of code that uses `||` (or) to determine whether the player wins a game of Craps

```
if ( roll == 7 || roll == 11 ) { // 7 or 11 wins on first throw
    status.setText( "You win!" );
} else if ( roll == 2 || roll == 3 || roll == 12 ) { // 2, 3, or 12 loses
    status.setText( "You lose!" );
} else { // Set point to be the roll to be made
    status.setText( "Try for your point!" );
    point = roll;
    ...
}
```





Nesting

- Suppose we want to decide among several choices based on several conditions, such as shown by the table:


	Sunny	Not sunny
Rich	Outdoor Concert	Indoor Concert
Not Rich	Ultimate Frisbee	Watch TV

- To do this, we use conditionals inside a conditional. This is called nesting.
- 



Nesting Example

```
if ( sunny ) {  
    if ( rich ) {  
        activityDisplay.setText( "outdoor concert" );  
    } else { // not rich  
        activityDisplay.setText( "play ultimate" );  
    }  
} else { // not sunny  
    if ( rich ) {  
        activityDisplay.setText( "indoor concert" );  
    } else { // not rich  
        activityDisplay.setText( "watch TV" );  
    }  
}
```



Instance Variables

```
public class WhatADrag extends WindowController {
```

```
    private boolean boxGrabbed; // Whether the box has been grabbed by the mouse
```

```
    private Location lastPoint; // Point where mouse was last seen
```

```
// Save starting point and whether point was in box
```

```
public void onMousePress( Location point ) {
```

```
    lastPoint = point;
```

```
    boxGrabbed = box.contains( point );
```

```
}
```

```
// If mouse is in box, then drag the box
```

```
public void onMouseDrag( Location point ) {
```

```
    if ( boxGrabbed ) {
```

```
        box.move( point.getX() - lastPoint.getX(),
```

```
                point.getY() - lastPoint.getY() );
```

```
        lastPoint = point;
```

```
    }
```

```
}
```

Instance variables are used to

‘remember’ data between methods

Assignment statement: ‘assign’ a value to a variable



Instance Variables vs Local Variables

```
private boolean boxGrabbed; //boolean variable to determine whether the box is grabbed
```

```
// Save starting point and whether point was in box
```

```
public void onMousePress( Location point ) {  
    lastPoint = point;  
    boxGrabbed = box.contains( point );  
}
```

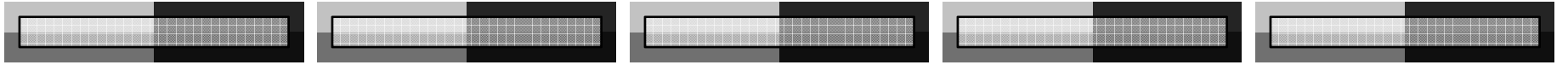
Instance variable

```
public class StickFigure
```

```
{  
    ...  
    public void move (Location point){  
  
        double headLeft, headTop, bodyLeft, bodyTop, leftArmLeft, armTop;  
        double rightArmRight, legTop, leftLegLeft, rightLegRight;  
  
        headLeft = point.getX();  
        ...  
        head.moveTo(headLeft,headTop);  
        ...  
        leftLeg.moveTo(leftLegLeft, legTop);  
        rightLeg.moveTo(rightLegRight,legTop);  
    }  
}
```

Local variables





Primitive Data Types

- Value can be written as a literal

Ex: `int a=1732`

- Can perform operations using operator symbols

Ex: `x+1`





Operators vs. Method Invocations

- Operators produce new values

Ex: if the value of `count` is 3

`count+1` will produce 4, but will not change **`count`**

If we want to change the value of `count`, how can we do it?

- Method Invocations can modify objects

Ex: `box.move(-1,-1)`; changes the position of the box





Operators and Precedence Rules

- Arithmetic and logical negations, - and !
- Multiplication and division: * and /
- Addition and subtraction: + and -
- Comparisons: <, <=, >, >=
- Equality and inequality: == and !=
- And: &&
- Or: ||

Ex: $a+b*c = a+(b*c)$ but not $(a+b)*c$





Numeric Types

- Integers (positive or negative whole numbers):

Ex: `int anInt=99;`

- Real numbers (contain fractional part):

Ex: `double aDouble=98.6;`

- An int can be converted into a double without loss of precision, but not vice versa
 - e.g. `anInt => 99.0` as a double
 - `aDouble => 98` as an int





Dividing int and double

The following table summarizes the types of result one will get when dividing two integers, two doubles, or a double and an integer

	int	double
int	int result	double result
double	double result	double result

Clearly, unless an integer is divided by another integer, all results are double.

Ex: $3.0 / 4.0 = 0.75$ $3 / 4.0 = 0.75$, But $3 / 4 = 0$! For ints, fraction is dropped.

```
private static final int FACE_SIZE = HEAD_SIZE * (4/5);
```

WARNING! FACE_SIZE is not 80% of the HEAD_SIZE!

4 / 5 is integer division and results in an integer, so 4 / 5 is 0!

Solution: Do the division last

```
private static final int FACE_SIZE = HEAD_SIZE * 4/5;
```





How a double is displayed

- If you print a double, the output is always accompanied by a decimal place.


Ex: `double a=1000;`

`System.out.print(a);` will output `1000.0`

- Large numbers use scientific notation

Ex: `double a=1000000000;`

`System.out.print(a);` will output `1.0E9`





Selecting a Numeric Type

- Use double instead of int whenever possible
- Use int when methods demand it


Ex: setColor(int, int, int)





Useful Functions on double

<code>Math.pow(a,b)</code>	a^b
<code>Math.exp (b)</code>	e^b
<code>Math.ln(a)</code>	$\ln(a)$
<code>Math.sqrt(a)</code>	Square root of a






String

- Java uses String to manipulate textual data
- Quoted text is of type String
- Programs can combine String objects

Ex: `String a="He";`


`String b="llo";`

`System.out.print(a+b);` // will print out Hello





Classes

- Part of a program
 - Template from which to create new objects, combines:
 - Data (features)
 - Methods (behaviors) to manipulate data
 - Encapsulates all features (all relevant info is in 1 place)
 - We've used many classes:
 - Location
 - getX, getY
 - FilledRect
 - getHeight, setHeight, getColor, setColor, move, moveTo, ...
 - Color
 - getRed, getBlue, getGreen
- 



Classes in Java

```
public class Basketball extends WindowController {
```

```
    private FramedOval hoop;  
    private FilledOval ball;
```

Data

```
    ...
```

```
    public void begin() {
```

```
        ...
```

```
    }
```

```
    public void onMousePress (Location point) {
```

```
        ...
```

```
    }
```

parameter

Methods

```
}
```





What if...

We can free ourselves from the limitations of ovals and rectangles and move on to...

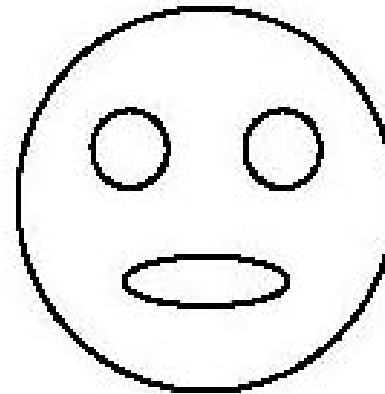
Creating our own classes. Consider making a funny face:

● Physical Characteristics:

- Head
- Mouth
- Two Eyes

● Behaviors

- Can check contains
- Movable (by dragging)





A Draggable Face

```
public class RevFaceDrag extends WindowController {
    :
    private FunnyFace happy;           // FunnyFace to be dragged
    private Location lastPoint;

    private boolean happyGrabbed = false; // Whether happy has been grabbed by the mouse

    public void begin() { // Make the FunnyFace
        happy = new FunnyFace( FACE_LEFT, FACE_TOP, canvas );
    }
    public void onMousePress( Location point ){
        lastPoint = point;
        happyGrabbed = happy.contains( point );
    }
    public void onMouseDrag( Location point ) {
        if (happyGrabbed ) {
            happy.move( point.getX() - lastPoint.getX(),
                       point.getY() - lastPoint.getY() );
            lastPoint = point;
        }
    }
}
```

Looks just like our
box dragging code.





Window Controller

public class BestBasketball extends WindowController

“extends WindowController” allows us to write a class that provides mouse handling capabilities
e.g. onMousePress, onMouseRelease, onMouseDrag, ...

“extends WindowController” also allows us to write the begin method that builds the initial world

“extends WindowController” lets us refer to the “canvas” and call the “resize” method

Classes that do not “extend WindowController”
cannot define mouse event handlers!

And cannot use canvas without declaring it!

