Conditional Statements, Strings, Fields.

In this lecture, we add another feature to GeometricFrame, and on the way, we learn about fields, conditional statements, and the String class.

Review from last time

Some questions asked at the beginning of lecture:

What is the purpose of this?

this is the address of the object the method is operating on. When we call a method:

f.transpose()

we are asking for the method transpose to act on the object whose address is stored in f.

When transpose is called, Java takes the address stored in f and places it into "this" so that we have access to that object inside the method.

There are two types of statements in Java.

1. All simple statements end in a semi-colon.

2. All compound statements (also called "blocks") are bracketed by { and }. Compound statements can contain 0 or more statements.

In this class, we create the method displaySizeOnTitle to display the window size on the title.

Examining the Java API for JFrame, we find the methods

void setTitle(String title) -> sets the title of the window to the input title

String getTitle() -> returns the title of the window

To use these methods, we need instances of the String class.

String:

String is a pre-defined class in Java and one of the most important classes.

Strings represent text as a sequence of characters.

Strings have a special means for creating instances. While we can create instances using the new operator (just as with any class), we can also just place the desired string inside double quotes.

"Hello" <- this is a shortcut to creating a new String using the appropriate sequence of characters as input.

Note that "Hello" acts just like the new operator, it evaluates to the address of the String object storing h,e,l,l,o.

We can use the object just like any other object. We can store it

String s = "Hello";

We can call methods on it:

s.length()

"Hello".length()

IMPORTANT: String instances can not be changed once created.

+ Strings have a special operator, the + operator. The result of + is a new String that concatenates the two operands together.

If one operand is not a String, it is converted (through a sequence of method calls) to a String before the concatenation.

Ex: "Hello" + "there" -> "Hellothere"

"Hello " + "there" -> "Hello there"

"x = " + 3 -> "x = 3"

Note: you must be careful when mixing Strings and numeric primitives with the +. When is the + meant to be a String concatenation and when is it meant to be a normal addition?

"x = " 3 + 5

First attempt:

public void displaySizeOnTitle() {

this.setTitle(this.getTitle() + " " + this.getWidth() + "x" + this.getHeight());

}

Works great, but if we call it multiple times, we keep concatenating the size.

Let's try again, but this time let's remember the original title.

For that, we need a field.

Field declaration (just like a variable declaration, but adds an access modifier)

private String originalTitle;

Why a field and not a normal variable?

3rd law of variables: A variable exists from when it is created to the end of the compound statement it is in.

If we made originalTitle a variable inside the method, the variable would cease to exist when the method was done executing, and the value remembered would be lost when we run the method the next time.

By storing it as a field, the variable exists as long as the object exists.

We made the field private so that it can only be accessed inside this class. That way outside code can not break the behavior of our class. (This is one reason for using private, but not the only one!)

When do we store the original title in the field?

Inside our method, right before we call setTitle to add the size to the title.

However, we only want to store the original title if we have not yet stored the title (otherwise we will store a title that contains the size!)

We need to set the original title to a special value that indicates it does not store a String.

null: a special value in Java that means "not an object" or "not a valid address"

We also need a conditional statement so that we only save the title if the originalTitle field is currently storing null.

Conditional statements

if (condition)

then-statement

else

else-statement

condition is an expression with a boolean type

then-statement and else-statement can be either simple or compound

The "else else-statement" part is optional.

How it works:

a) the condition is evaluated

b) if the condition is true, the then-statement is executed

c) if the condition is false and the else-statement exists, the else-statement is executed

Here is our next attempt:

private String originalTitle = null;

public void displaySizeOnTitle() {

if (this.originalTitle == null)

this.originalTitle = this.getTitle();

this.setTitle(this.originalTitle + " (" + this.getWidth() + "x" + this.getHeight() + ")");

}

Problem:

If we change the title of the window -after caling displaySizeOnTitle()- and call displaySizeOnTitle() again, the original title, not the new one, is used.

Solution? Stay tuned (or see if you can come up with one on your own.