Methods, Conditional Statements, Designing Code

Last time we wrote the following method and field to add the window size to the title of our GeometricWindow

private String originalTitle = null;

public void displaySizeToTitle() {

if (originalTitle == null)

originalTitle = this.getTitle();

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

}

How does it work?

originalTitle stores the title that the user (or whoever) wants to place on the window.

We start originalTitle to null (meaning "not an object") to indicate that we do not have anything stored.

The first time we call displaySizeOnTitle(), originalTitle will be null, and so we place the address returned by setTitle() into originalTitle.

After this time, originalTitle will be storing the address of a String and it will not be null.

Finally, we create a new String containing the original title and the window size and make that the new title of the frame.

Note this does not work in all cases that we might want:

1) We can't remove the size from the title (other than by setting a new title)

2) originalTitle never changes from the first time we call displaySizeOnTitle()

Let's deal with (1) first. The class vote was to change displaySizeOnTitle so that it takes a boolean that specifies whether we should add or remove the title from the frame

Let's choose a good name for the input. The first option was to use visible as the variable that stores the input

public void displaySizeOnTitle(boolean visible) {

if (visible == true) {

if (originalTitle == null)

originalTitle = this.getTitle();

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

}

else { // do something to remove the title

}

}

However, the line

if (visible == true) {

is not the best form. The expression is true if visible is true and false if visible is false, so we could just use the shorter

if (visible) {

The new form is better Java, but it does not read as well in English. So, let's choose a better variable name for the input so that it reads more closely to how it works.

We are going to display the size if the input variable stores true so we decided to use:

if (turnDisplayOn) {

And so the whole method is now:

public void displaySizeOnTitle(boolean turnDisplayOn) {

if (turnDisplayOn) {

if (originalTitle == null)

originalTitle = this.getTitle();

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

}

else { // turning the size display off

}

}

Next, we have to figure out how to remove the size from the title. Once it was observed that the originalTitle field stores the title without the size, removing the title is just restoring the title to the original title:

public void displaySizeOnTitle(boolean turnDisplayOn) {

if (turnDisplayOn) {

if (originalTitle == null)

originalTitle = this.getTitle();

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

}

else { // turning the size display off

this.setTitle(originalTitle);

}

}

However, we only want to set the title to originalTitle when originalTitle actually stores the original title (or technically, the address of the string of the original title). If originalTitle stores null, then calling setTitle(originalTitle) will

incorrectly remove any title currently on the window.

public void displaySizeOnTitle(boolean turnDisplayOn) {

if (turnDisplayOn) {

if (originalTitle == null)

originalTitle = this.getTitle();

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

}

else { // turning the size display off

if (originalTitle != null)

this.setTitle(originalTitle);

}

}

Now let us deal with (2):

The problem is that originalTitle is only set once. If we think about it, we only need to store the original title while the size is displayed. When the size is not displayed, we can clear out the original title so that the user can freely change the title:

public void displaySizeOnTitle(boolean turnDisplayOn) {

if (turnDisplayOn) {

if (originalTitle == null)

originalTitle = this.getTitle();

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

}

else { // turning the size display off

if (originalTitle != null)

this.setTitle(originalTitle);

originalTitle = null;

}

}

The result, the method works as long as the user calls displaySizeOnTitle(false) before changing the title.

We need two more fixes that will be done in future lectures:

(1) If the title is changed while the size is displayed, allow the change and keep the size displayed.

(2) If the size is changed while the size is displated, automatically update the size displayed.

One last method example:

Let's create a method that takes two windows and returns true if the first is larger than the second.

What shall we call it? The class decided on: isFirstLargerThan

Does the method act on any GeometricFrames other than the two input? No, so this should be a class method instead of an instance method.

Do we need the inputs to be GeometricFrames? No, we can use JFrames as the inputs (or maybe even Frame or Window). By polymorphism, any GeometricFrame is also a JFrame so setting the input type to JFrame increase the different types that will work.

How do we compare sizes? The class decided on using the area of the JFrame:

public static boolean isFirstLargerThan(JFrame frame1, JFrame frame2) {

return frame1.getHeight() \* frame1.getWidth() > frame2.getHeight() \* frame2.getWidth();

}

Remember that to call a class method, we do not need to place an instance address before the dot. The classname also works before the dot:

GeometricFrame.isFirstLargerThan(new GeometricFrame(), new GeometricFrame())