Transcript: Die Class

>> Okay, let's take a few moments and review the die class. Recall that it contain two fields or data declarations, a constant MAX, that represents the maximum face value and then also, an integer face value that represents the current face value. And, and that's the instance variable in this case.

A class could have lots of instance variables, usually, it, it does have several. But in our case, we have one instance variable. Then we had a roll method that used the random method of the Math class to determine a new face value. And then there were methods to explicitly set and retrieve the current face value any time a calling program needed it.

Also there was a toString method. Now in general, all classes that represent an object should declare a toString method. And it should describe the object in some useful way, in our case it just described the face value. That that's really all we had about a die, and so it returned the face value.

But keep in mind this two string value, or two string method rather is called any time an object is referenced where a string is needed. So that would be in a system, out.printline if you concatenated it with other things it's gonna be a string or if you're just printing it, it would be a string.

And then also in interactions, if you keyed in the variable name say die1, if die1 was active you would get the value of die1. We also had a constructor, it was a simple constructor in our case. In general you need a constructor that is called using the new operator to create an object and usually set its initial state.

If you're not setting it directly in the constructor, it'll receive default values and int would receive a zero and if you had a reference type such as a string as a field, it would be set to a null. But keep in mind it has the same name as a class and no return value.

If we look at our constructor for die, we see that again, same name as a class, no return value, very important. And all it's doing is initializing the one instance variable. Constructor may or may not have parameters, ours didn't. If you needed more than one constructor, maybe you only have one that takes parameters you can certainly have those.

It was just, all, how many ever constructor's you have, they have to differ by number and type and order of parameters. Again, in our case, we had zero parameters. And, and it did set the face value that that´s, and that´s a common thing for a constructor to do to assign the initial values.

Now, let´s talk about scope of variables, that´s, that´s really just where a variable can be referenced. It´s determined normally where the variable is declared, that is it´s enclosing block and if we look at instance variables, they're declared, at the top level. Notice they're inside the, directly inside the block of it, for the body of Die.

And so they're actually visible to anything in that class. And and which includes all of the methods they can get to the instance variables or those fields and directly and so on. If we have Local variables, a local variables defined as one declared inside of that method. It can only be referenced in that method in fact indeed it comes in to existence during the execution of the method and when the method ends it goes away.

Example of this is the variable result into string. Let's take a quick look at it here its is right here. So, when two string is called string result is declared and it said equal to the string value of face value. And then that's returned from the two string method, it's got a string type is the return type, so it has to return a string, and then as soon as it hits the closing brace of course result goes away.

So the only place result is in scope is inside the method there. More about instance data now. We, we, we hit on this. But it's worth, worth repeating. Each each instance of the die class has its own version. And the declaration of an instance variable declares the type.

In our case, we had a an int. But it doesn't reserve, any memory space for it, when a Die object is created using the new operator, that's when the, the faceValue variable is created. And and also gets a value, in our case the constructor, set it to one, it could have taken in a parameter and set it to something else, but in our case it just set it to one.

Now, each object has its own set of instance variables and in our case, that means a faceValue. But all of the objects that are created, in our case, Die one, Die two, and so on, they share the same methods. So, so the object has its own data space for the instance variables, but it shares all of the methods in the class.

Now looking at the instance data, here's a a graphical depiction here. Die1 is a reference variable and it points to the object of a die, a die object that was created. In, in, in this particular case, the faceValue of die1 was 5. And here's die2, and it's a reference variable.

It points to a die object, and its face value is 2. And as we saw in jGRASP, it depicts the same scenario as follows there. There's die1 with a face value of 5 in die2, with a face value of 5.