Introduction to data types and variables

Video transcript
https://www.futurelearn.com/courses/begin-programming-2014/steps/7624/progress

[MUSIC PLAYING] Numbers aren't just numbers. We know this. They refer to something else. For instance, when I was a kid it was very important for me to know how many cakes I'd eaten, because I could not live with my brother having more than me.

In maths we also start using numbers for different things. For instance, we could put a dot in the graph, and having the x and the y value, suddenly, we would be able to create new worlds in graphs. We use this a lot in programming.

With variables, we give them a name so that they become meaningful to us. For instance, we will have a variable that denotes where the ball on the screen is in the x direction, and where the ball is in the y direction. We have a name for this variable, and therefore you know what it does. Yes, there's a number in the background, but the variable is the important part.

We have many different data types to work with-- many different kinds of numbers, many different kinds of things that we can use the variables for. We'll look at that later on.

So let's have a look at how it looks in the code **(Looks at game code in Eclipse)**. If we go into the code, we can see that we're using five variables. Fire variables exist that we are using inside our programme to control the ball in the middle of the screen. As in many things, it's easier to start from the end. So let's do that **(Looks at '//The X and Y position of' in the code)**.

The ‘mBallX = 0-‘, all that says is that we have a variable, which we call mBallX, and it's equal to 0 to begin with. ‘mBallX’--, well the ‘m’ is a convention that we're using inside Android programming. It actually stands for member, ‘m’.

It's easier for us to find them later on, when we have many of these variables. So make an effort and put this m in front of your variables when you create new ones yourself. Ball indicates that it has something to do with the ball. And ‘X’ is the ‘x’ position on the screen. Likewise, ‘mBallY’ signifies that it's the ball's position in the ‘y’ position of the screen.

‘MBallSpeedX’ indicates that it's the speed of the ball. How many pixels per second will this ball go, in the ‘x’ direction? And likewise, ‘mBallSpeedY’ is the speed in the ‘y’ direction.

The private part, we forget about. It's something to do with a very advanced aspect of programming. It's called object oriented programming, what we're doing here. If you are very interested you should go and look it up later on, after you've done the course. It's actually quite exciting. And there are many nice features that we can use with these names. But for now we just forget about it. Just remember that we have to put it there.

The float is what type it is. It signifies that the ‘x’ and ‘y’ of the ball is a float number. A float number is a number with a dot in it. So we can say ‘10.2’, or ‘10.5’. Likewise, the speed is also a float.

Let's see how it works. I have started up the emulator, and have it running here **(Opens emulator)**. What I will do here is use the debugger. The debugger is a very nice feature inside Eclipse. It shows us what's going on in the background. It's very nice for finding problems with the code. So if you've made a mistake we can go in and look at the code and see what, exactly, happens. It's always very nice to see what is actually happening and understand what is happening.

So first of all, what I do is I put in a breakpoint here **(Next to 'setupBeginning' code in Eclipse: Double-clicks to insert breakpoint)**. You put in breakpoints by double clicking on the code. So we have a breakpoint there. We're going to be setting up the variable, the first time, in the setup. Beginning method. We'll get to what a method is later.

But this is the code that we'll be running when we start the game. So when you click Ready, all the code inside ‘setupBeginning’ will be run. So I set up a breakpoint there, so we'll see **(Scrolls down to 'actionOnTouch' code)**.

I've also set a breakpoint in the code that is run when somebody touches the screen. And that happens down here, in ‘actionOnTouch’. So when somebody touches the screen, this method here gets called. Again, we'll talk about what methods are later on in the course. But this code in here will be run.

So let's put a breakpoint here, like so **(Double-clicks to insert breakpoint)**. Double-click, and nothing happens. There we are. No, it happened. Sometimes it can be a little bit slow, especially when making a video. OK. So this is where the code runs when we press the touch screen. If you are running it from a phone and want to see what happens when a change happens on the phone it's down here, in what I have commented off.

It's really difficult to set a breakpoint there. Because it would literally happen every single time you move the phone. And you move the phone all the time. So it will break– it will stop the code, sorry– every single time somebody does that. So be a little bit careful with doing that.

Let's start up the debugger. Do it up here. And we just start it instead of pressing the Run button **(In icon menu bar: Clicks [debugger] icon > [1 uk.ac.reading.sis05kol.mooc])**. Let's bring up the emulator **(Opens emulator)**. It takes a little time to bring up the app. As you can see, a lot of things are happening in the logcat. Here we are. And voila, we are ready. And I press the middle screen **(Clicks green screen)**.

And you see something came up here. Confirm Perspective Switch– it's to do with the different aspects of Eclipse **(In 'confirm Perspective Switch' dialog: Clicks [Yes])**. What we have seen before was where we develop. Now Eclipse asks us, do we want to go into the debug perspective? And yes, we do. Here we are.

So now the code has stopped. And to go through the code, we could do it step by step **(In the 'TheGame.java' tab in 'Debug' dialog: Highlights 'mBall = mCanvasWidth / 2;') > ([F6]) > (Shows event in [Java] mode)**. We can use F5 or F6. F6, for instance, would just take and execute one line at the time. We can also go up here and look at the Run. And then we can use the 'Step Into', 'Step Over', and 'Step Return' **(In 'Debug' dialog menu bar: Shows [Run] > [Step Over])**. The one that I usually use is Step Over. And we'll talk about what the others mean later on.

So we use the 'Step Over', so F6. So I do F6. And let's see what happens to 'BallX'. You can find 'BallX' under this **(In 'Variables' tab: Clicks arrow next to 'this' > shows variables and values)**. And here we list all the different variables that we're using. There are many here, because that's a lot of code that we don't touch. But because we started with them, they're easy to find, the ones that are important to us.

And here we see the five different ones that we created. And now, as you can see, 'BallX' is actually equal to 240, instead of 0, as we thought it would have been. And that's because we have 'mCanvasHeight', which we have been given from somewhere else, from the code that has been delivered to us.

And we can see that they have already got values here. So the height is 682, and the width is 480. And when we go through the calculus, we say 'mCanvasHeight' divided by 2 **(Looks at code in 'TheGame.java' tab) > (Returns to 'Variables' tab)**. Then, we'll get the height in the 'mBallY'. So we'll expect to put the ball in the middle of the screen. Let's try again with y **(NOT SHOWN- Moves ball in emulator)**. Here we are. It changed.

So now we have the position in the middle of the screen. So we expect, now, to see the ball in the middle of the screen. Let's keep it running. We do that by pressing the Resume button, up here **(In icon menu bar: Clicks [Resume] button)** . And let's go in and see **(Opens emulator)**.

Oh, there we are. Now we have the ball on the screen. The next thing we could do is press the button– sorry, press the screen **(Clicks green screen)**. And we'll have the ball move. So if I press it here, voila **(In 'Debug' dialog: Shows code at breakpoint)**. We stopped because we had a breakpoint.

In the breakpoint something should happen. So we expect the ball to have a value. And if you hover over it you can see that we get the value **(Hovers cursor over code)**. We don't have to go into this to look at it. And it's 240. That's where we have the ball at the moment. The new ball speed should be x **(In 'Variables' tab: Shows new value)**. And x, you can see up here, is 294. And y is 231. And that was the position where I clicked on the screen.

So what I say here is that I'll give it a speed depending on where I clicked, and where the ball is now **(In 'TheGame.java' tab: Hovers over 'mBallSpeedX')**. So if the ball is far away from where I clicked, the ball will go fast. If it's close to, it will go slow. So now the speed should change. And right now, it's 0. So if we go one step, I click F6 **([F6])**. Now hover over it. There we are. It's now 54.

So now the ball should move in the x generation with 55 pixels per second **(Hovers over 'mBallSpeedX')**. And y likewise changes once we do the line, to minus 110– so quite fast in the y direction **(Hovers over 'mBallSpeedY')**. Let's keep it running **(In icon menu bar: Clicks [Resume] button)**. And we'll see the ball move if I'm fast enough. I wasn't **(Opens emulator)**. Let's remove the breakpoint so we'll see it actually running **(In 'TheGame.java' tab in the 'Debug' dialog: Removes breakpoint) > (In icon menu bar: Clicks [Resume] button) > (Opens emulator)**.

I might have to restart the code here. I'll just do that. We'll restart it without doing any of the debugging so we can see that it works **(Clicks: [restart] button) > (In icon menu bar in 'Debug' dialog: Clicks [Run] button)**. That's the problem with sometimes doing the debugging and filming it **(Opens emulator)**. Here we are **(In 'Debug' dialog: Shows breakpoint and icon menu bar)**. We have the breakpoint in the setup, so we expect everything there to work. Here we have it, on the screen **(Opens emulator: Clicks green screen)**. And now the ball moves and I can keep it moving **(Moves ball on screen)**. And we could play around with that for a long time. But I won't, so let's stop it **(Minimises emulator)**.

Many types exist. So in this instance, we were using floats **(In Eclipse: Returns to [Java] mode)**. But as you'll see in some of the text, there are quite a lot of different types that we can use. We can use Booleans, integers, strings, and many others. And if you remember, there was one I left—‘mBall’. ‘mBall’ is a bitmap **(Looks at 'mball' in code)**. It's an image. It's an image that you will actually see later on, that we can take from over here, and bring into our code **(Looks at 'drawable' folder in Eclipse package explorer)**.

There are many different types. And some of them are even more advanced than the bitmap. You could actually put in a browser– a browser window– into an app. So they're very advanced. And the possibilities are endless with these. So there are very advanced effects, features, and functionalities that we can bring in to our app once we know how to do the coding. This indicates that.

If we go down to the code when we start up the game, here **(Scrolls to and highlights 'TheGame' code)**, this code is actually how we say that we want the small ball from over here in this folder in our code **(Looks at 'drawables folder')**. So if you wanted to change it, you could just change the small red ball to, for instance, being a smiley ball, if you thought that would be fun.

You will use variables throughout the course, so learn more about them in the next step.