TKP Lesson 1:

Objects, Methods and For Loops

**Preparation**

* Have SimpleSquareWorksheet printed (see Step 6)
* Have Eclipse set up on computers
* Install from GitHub – TeachingKidsProgramming. (lfal.co/tkpjava)
  + Download zip link on right. Hit the launcher.
  + If it doesn’t open at all, need to install Java.

**During class**

Kids all working in pairs unless otherwise noted. 5 minute timer – kids switch off. One types, while the other tells the typist what to do.

# Part 1 - Recipe: Simple Square

Overall process:

* Have them read the line out loud.
* Guide them to figuring out the code themselves, to whatever extent possible.
* Once they figure it out, MAKE SURE THEY RUN THE CODE BEFORE PROCEEDING. (Use <http://virtualproctor-tkp.appspot.com/> to show student windows on screen when they close)
* Teacher notes
  + Red box will kill the program.
  + XX will reset the program, if only ONE is running. (useful if machine is being weird.)

LINE NOTES

* #1: Tortoise.show();
  + Ask: what is the noun? Tortoise.
  + To get it to do something, “.” after, and then our command.
  + Show them ctrl-space. Be lazy!
  + We end sentences w/periods in English, but w/semicolons in programming.
  + Capitalization matters. Have them test with lower case.
  + AS YOU GO, clean up the English. Command-D.
* #2: Tortoise.move(50);
  + Remind them to clean up the English
* #3: Tortoise.turn(90);
* #4: Tortoise.*setPenColor*(Colors.Blues.*blue*);
  + Explain setters and getters.
  + Setters change something about an object; getters tell us something about an object.
    - Choose a student to be an object.
    - Set a student’s position to standing.
    - Get a student’s hair color.
* #5: **for** (**int** i = 1; i <= 4; i++)
  + Have them read the text (***// Do the following 4 times --#5.1***)
  + Ask for what needs to be repeated. Have them phrase it in terms of line numbers.
  + Easiest way – copy and paste four times.
  + But what if you had to do it 400 times? Violate the most important rule for programmers – be lazy. We don’t want to break that rule. So…
  + For this one, show them to type “for” and hit ctrl-enter. Sets it up for them.
  + “Stop” is in red, so needs to be replaced.
  + Keyboard shortcut: alt-up or down, to move end brackets to bottom of loop.
  + Make sure they test. Check – easy to mess this one up!
* #6: Tortoise.*setSpeed*(10);
  + After eliciting the command, ask: how do we know what is “as fast as possible”? Ensure they’re reading the text that comes up.
  + Have them try it.
* CONGRATULATIONS! You just made your first program! ☺

# Part 2 – Recap: Simple Square

[**http://lfal.co/RecapSimpleSquare**](http://lfal.co/RecapSimpleSquare)

# Part 3 – Variation: Simple Square

* Open Word or something to scribble in. Make a grid
  + Add columns – feature, value
  + Elicit from kids – what features are there?
* Example (with all later steps completed):

|  |  |
| --- | --- |
| **Feature** | **Value** |
| Angle | 90 |
| Color | Blue |
| Width | 2 |
| Sides | 4 |
| Length | 50 |

Preparation

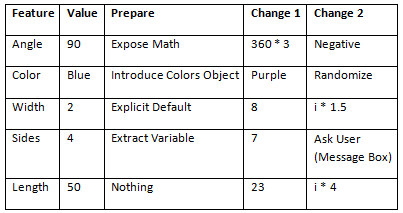
* Intro to preparation
  + Any of these values can be changed. But first, we have to get the code ready to be changed, or it’ll end up a mess!
  + [Add another column – prepare. Must do these before moving on to other things.]
* Prepare angle
  + Hard to change right now
    - Go to 90 in code. 90 = quarter of circle. (Spin around.)
    - How do we know that 90 is ¼ of 360? Math. (Or memorization.)
    - Computers better at this than us. What if we want it to turn 1/7 of a circle? What is 360/7? Uhh…
    - Rewrite as 360/4
    - Run, see that it’s the same. [THIS IS IMPORTANT.]
    - Put X in “prepare” for angle.
* Prepare sides
  + Simple refactoring. Go to where it says 4, highlight, right click, refactor, extract local variable.
  + Name it “sides”
  + Run – show that it does exactly the same thing.
  + [Don’t explain everything here – just show that it runs.]
  + X it off in the graph.
* Prepare line width
  + Which line determines this? None. It’s a default. So we’re going to make the default explicit.
  + FIRST LINE IN for loop, add: Tortoise.setPenWidth(2);
    - Go back to grid, X it off

Variations

[Overall note for variations: try 3-5 numbers, some ridiculously small, some ridiculously big. Don’t use “normal” numbers (20, 100) – use 13, 117, etc.) Don’t go over 1000.]

* **New column: “Simple changes”**
  + **Add one more column: “Cool changes”**
  + Can change anything, but needed to prepare before we mess around.
* **Variation: sides**
  + Change sides to 7.
  + Change sides to 13.
  + Change sides to 100.
  + Anyone notice a problem?
    - Lines don’t meet at a certain point. Why? What is 360/13? Fraction!
    - Java doesn’t automatically deal with fractions… so change 360 to 360.0 so we’re telling it we’re dealing with fractions.
* **Variation: colors**
  + No refactoring – show green, purple, let them pick a couple of favorites.
  + Show swatches of color as they arrow down.
  + Have them find a way to get a random color
    - Tortoise.*setPenColor*(Colors.*getRandomColor*());
* **Variation: line width**
  + Start with a few small – 5, 8, 13, 75, 347, bring it back to 7
* **Variation: Move length**
  + 99, 9, 23. Check it off. (Just quick changes here.)
* **Cool changes**
  + Every time we run this it does the same thing. Now, let’s make it so user tells us how many sides to include.
  + Replace sides = 7 with messagebox.
    - int sides = MessageBox.askForNumericalInput("How many sides?");
    - Run it – show a few options
    - Show that sides is in black. So is i. These are variables.
    - In “setPenWidth”, change 1 to “I”.
    - Ask: what do they think will happen?
    - Talk them through what’s happening. How wide is line 1? What about the fifth line? The (next one after the largest?) – there isn’t one.
    - Do a few more – I \* 10, I \* 3.5
* **Variation: move**
  + i\*5
  + run it
  + Ask about shape: if those were connected, it would be a circle, right? That’s how many degrees? 360. So if you wanted three circles, how many degrees would you need? (Some student says 1080. “You violated our first rule – let the computer do the math!” So: 360\*3.)

Put up virtual proctor on screen and let them play with it, seeing what everyone’s doing.



# Part 4 – Quiz:

SimpleSquareQuiz.java

* This takes about two classes. (About an hour, but with transitions…)
* Cmd-shift-T: SSQ (SimpleSquareQuiz)
* Rules for quiz
  + Kids on own, but you show them how to do it.
    - They simply enter code matching recipe, then run and it will tell them if they passed the test.
  + Run after each question *before deleting recipe line*.
  + **How to help kids who get stuck**
    - Don’t show – suggest how they can discover for themselves.
    - E.g.
      * “Read the English again”
      * “That method doesn’t seem to be working for you – maybe try another one.”
      * “You haven’t gotten question 1 working. Let’s undo and get that working before doing question #2.”

# Part 5 – Deep Dive:

Homework01.java

* cmd-shift-t, type “home”
* Show two videos first to give context
  + <http://www.youtube.com/watch?v=vJG698U2Mvo>
    - If they have seen it, you can use this one: <http://www.youtube.com/watch?v=IGQmdoK_ZfY>
  + The point: no matter what happens, you’re going to miss a lot of it. So one way to counteract it is to get everyone to volunteer what they see, even if they think it’s very obvious. (Could be as obvious as a gorilla walking across screen, since someone might not have seen it!)
  + Can also show McGurk effect.
* SETUP
  + Chairs at front, circle/oval/whatever – computer at desk to the side.
  + Give kids post-its.
  + Each kid writes an observation and the last thing they just did. No talking – they just write. Then collect, and read them to the class.
* PROCESS
  + When going through Deep Dive, one student will sit @ computer, another will be standing and will tell them what to type.
  + After each test, standing student gives observation (/explains why they entered what they did), then they rotate.
  + If they ask “Does it work if…” then try it and see!
* SHOW THEM THE PROCESS
  + numbersDoNotNeedQuotes
    - Put cursor on method name. Then click on “numbersDoNotNeedQuotes”, run.
      * Show that it does not work.
    - Walk them through getting it to work, then have them write observations on it. They read.
  + defaultWidthForTheTortoise
    - Show failure trace – how do we find the right #?
  + stringsNeedQuotes
    - Type green
      * Did not work. Why? Look at the title!
    - Change to “green”.
      * We done? Nope! Must run.
      * Double click on the failure trace when running the failed program – show the window that pops up.
  + theTortoiseTurns15Twice
    - How is this different from the last one? (Turning twice, but looking at the angle which results from both turns)
  + combiningNumbers
    - How is this different from the last one? (Adding numbers, looking for total)
  + combiningText
    - How is this different from the last one? (Adding strings just combines them one after the other.)
  + combiningTextAndNumbers
    - How is this different from the last one? (Adding a number to a string treats the number as a string.)
  + Useful techniques – asking how one is different from the last. (E.g., “assigningVariables” vs. “howFastCanTheTortoiseGo”
* combiningTextInALoop
  + Ensure the kids get what happened here. Maybe pick one from audience to explain.
* forLoopsEndAtTheLine
  + A fairly advanced thing
  + Double-click in blue thing. Debug dot.
  + Now, right click on for loop in left part… sends you to debugger.
  + Use “step over” (yellow arrow top right) to see individual steps. Show the kids how this works.
* forLoopsCanStartAnywhere
  + Same as above – have one explain
* forLoopsCanSkip
  + Same as above – have one explain

# Part 6 – Worksheet:

SimpleSquareWorksheet.docx

* Go to navigator, click fn-F5
* Under worksheets, go to “SimpleSquareWorksheet”. Have this printed. MIGHT NEED TO RIGHT-CLICK AND REFRESH PROJECT.
  + Have kids circle part of code, part of English, draw line between them.
  + Have them go through it – use teacher version for notes.

# Part 7 – Solo Recipe: Spiral

* Open “Spiral”
* Each kid is on a single computer.
  + [[Solo work is not a good way to teach. This isn’t teaching; this is exposing whatever individual misunderstands they might have.]]
* If kids finish early, they can try variations.