TKP Lesson 2:

Methods

**Preparation**

* Need worksheet printed?

**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: Houses

Process reminders:

* 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)
* Ensure they’re closing, not minimizing, the window.
* Teacher notes
  + Red box will kill the program.
  + XX will reset the program, if only ONE is running. (useful if machine is being weird.)
* 1.1: Tortoise.move(height);  
  1.2: int height = 40;
  + Noun; Connect with a period
  + What are we trying to do? How far do we want to move?
    - We don’t know, so we fake it.
    - When taking, good to use “crazy” number so you remember it’s fake.
    - So use 1000 and see if that works.
    - (Don’t delete English, we’re still faking it – not done yet!)
  + **Refactor**: refactor the fake and extract it as a variable, name it “height”.
  + We have now created line 16: int height=1000
    - But it is not translating line 15, so move the code to follow the English its translating and replace the fake number with the information we learn: 40
  + Delete English lines 1.1 and 1.2
  + This one’s already done now! Run to test, then English can be deleted.
* [For 2-8, do them quickly as a class. If they forget to turn numbers negative, just run it and let them be confused. ☺]
* 2. Tortoise.turn(90);
* 3. Tortoise.move(30);
* 4. Tortoise.turn(90);
  + Ensure they do this the lazy way! (And the following)
* 5. Tortoise.move(height);
* Make sure kids are using “height”, not “40”
* 6. Tortoise.turn(-90);
* 7. Tortoise.move(20);
* 8. Tortoise.turn(-90);
* **#9: sub-recipe**
* THIS IS WHERE WE GET TO SOMETHING TOTALLY NEW.
  + Anyone notice something different about this recipe? – It’s like a recipe within a recipe!
    - This is a sub-recipe. When you make a cake, there is the recipe for the whole cake and the recipes for the parts: cake, filling, frosting;
  + Show them how to extract a method. Highlight from beginning of method to end, INCLUDING comments for #9 (top and bottom). Choose Refactor 🡪 Extract Method, then use name of sub-recipe: drawHouse.
  + Delete #9s, run and see that it does exactly the same thing.
    - When we refactor, code should run exactly the same.
  + Have kids look at it and try to figure out – what did we just do? Why might this be useful?
  + Give terminology: *method*. Why would we do it? If we’re doing the same sort of thing over and over, and want to save typing. Lazy = good. ☺
* Make sure they delete #9!
* 10. height = 120;

drawHouse(height);

OR

drawHouse(120);

* + **Before typing –** can they see two ways to do this? (hint:one is two lines, one is one)
  + What might be advantages, disadvantages of each?
    - Height variable – can be changed more easily later. (E.g., if you want to ask user what height to use.)
    - 120 – less typing.
    - So a tradeoff, depends on what you want. For now, both do the same!
* 11. Tortoise.setSpeed(10);
  + Lots of houses to draw, so let’s get this tortoise moving!
* **Have them do 12-13 by themselves. They know how.**
* 12. drawHouse(90);
* 13. drawHouse(20);
* 14. Tortoise.setX(200);
  + What is the noun? Tortoise.
  + How are you going to translate this line? “set-x”
    - “Get” and “set” is easy mistake to make, happens all the time.
* 15. Tortoise.setPenColor(Colors.Grays.LightGray);
  + They know how to do this.
  + Run it!

# Part 2 – Recap: Houses

[www.youtube.com/watch?v=C6fnqjceVcs](http://www.youtube.com/watch?v=C6fnqjceVcs)

**Focus on creating methods**

# Part 3 – Variation:

VARIATIONS

* “Tell me the attributes of what we’ve made”
* Make grid – brainstorm features (necc. ones below)
  + Height of buildings
  + Width of building
  + Number of buildings
  + Color of line
  + Position of buildings on page
  + Number of sides on buildings
* Have the kids pick one to change, ask them where it is in the code.
  + Teaching point: if you see it, you can change it. If not, you can expose it, often by refactoring.
* Variation: peaked roof
  + Have kids identify what three lines make the roof, then where it is in the code. (So tortoise moves, turns, moves, etc.)
    - Tortoise.turn(90);
    - Tortoise.move(30);
    - Tortoise.turn(90);
  + Have them identify the shape. Is it interesting? Not really.
    - Call one out – they come up to board.
    - How would we make the roof not boring – i.e., pointy?
      * Have student identify – would have to make it go up and then down, like a triangle.
    - Two ways to do it
      * One pair comes up, driver/navigator style
      * Have them actually move physically to show the roof.
      * Have them draw it on the board, and then write English on the board to match.
        + During process, have them identify the # of degrees it needs to turn, and how far it needs to go after each turn.
        + (Can have them sit down during this and then class yells out English to make it.)
      * Then have them go to computer and try it out
    - OR
    - Refactoring
      * Which lines make the roof?
      * Highlight, right-click, extract method. Call it drawRoof.
      * Run to check that it still does the same thing
    - VARIATION: now we break the code
      * Let’s make it NOT do the same thing.
      * Go into the method, change some things, see that it’s now broken.
      * Challenge: who can make a flatter roof? Proctor on screen.
    - Other variations:
      * # of houses, colors, setX, etc.

# Part 4 – Quiz:

HousesQuiz.java

* Warn in advance: this one is complicated, and you will need to be comfortable with some trial and error.
* Remember, ctrl-Z to undo – your best friend!
* Hints, if needed:
* For question2, what’s missing? We need to add the question method itself!
  + How? Copy from above, and change
  + public void question2()
    - Make sure moustaches go around the English for the question

# Part 5 – Deep Dive:

DeepDiveVariables02.java

# Part 6 – Worksheet:

No worksheet yet.

# Part 7 – Solo Recipe: Four Square or DigiFlower