|  |
| --- |
| GAME- BASED TEACHING MATERIAL |
| Hug the line (HTL) |

# 

[[Lab 1] Method: What Does a Method Look Like?](#_kcv51a9q2flh)

[[Lab 2] Method: Drawing Wizards with Parameters](#_ig3lhiqoh3bj)

[[Lab 3] If/Else: Draw Wizards with If/Else Statement](#_kmplnl7b2syf)

[[Lab 4] If/Else: Using Variable Declarations](#_eerek4y3dgpe)

[[Lab 5] Nested If: Moving Wizards](#_ajipk4d5or59)

[[Lab 6] If/Else: Personalization](#_yr94wmzak69)

[[Lab 7] For Loop: Create Custom Paths](#_pdspdmjusjx1)

[[Lab 8] While Loop: What Is While Loop?](#_rjot4onmbrp8)

[[Lab 9] Nested Loop: Make Wizards Casting Spells](#_c9dl9x1x50lg)

[[Lab 10] Gameplay: Win Conditions](#_f6ho5kqed8e8)

[[Lab 11] Gameplay: Cleanup The Game!](#_8y46mbneajp1)

[Resource](#_xj3ywjbdvs7s)

# 

# 

# [Lab 1] Method: What Does a Method Look Like?

Activity

Concept:

The format and parts of a method/function in java.

Learning Objective:

Student will be able to write a method/function in Java.

How to Assess (optional):

Look at student output to verify their methods/functions compile and are working as desired.

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Tell the student that they will be learning how to write a method or function.
2. Explain to them what a method or function is:
   1. Most programs perform tasks that are large.
   2. Instead of writing a large program as one long sequence of statements, programmers usually break down their programs into several small manageable pieces known as method/function.
   3. Each method/function only performs a specific part of task.
   4. It can be executed in the desired order to perform the overall task. (Gaddis)
3. Give the students some real-life example that they can relate to such as:

|  |  |
| --- | --- |
| * 1. A Light switch      1. Turn on the light      2. Turn off the light | * 1. Game Character      1. To walk      2. To run      3. To jump |

1. Now, open BlueJ and go into the “MyGame” class.
2. Tell the students that the first step in writing a method/function is to identify the task you want it to perform.
   1. You want to build the world for your game.
3. A method/function need to have a name to be called.
   1. Should be descriptive enough so that anyone reading your code can reasonably guess what the function does (Gaddis).
   2. Naming method/function is similarly to naming variables, which we recap here:
      1. It cannot use one of existing Java’s keywords such as class, public or private, and void.
      2. It cannot contain spaces.
      3. The first character must be one of the letters a-z, A-Z, or underscore ( \_ ).
      4. The first character should be a lowercase letter known as camel case.
      5. Uppercase and lowercase characters are distinct. (Gaddis).

|  |  |
| --- | --- |
| * 1. Type the name of the method/function on the board following by open and close parentheses. | **Note:** You will need to name it based on what it is named in the API. It will be buildGame(). |

1. Every method/function has an access modifier:
   1. Access modifier is a keyword that specifies accessibility of the method/function

|  |  |
| --- | --- |
| * 1. There are 2 main modifiers:      1. *Public*: method/function is accessible everywhere. In other words, codes from other places can use it.      2. *Private*: method/function is accessible only within class. In other words, only codes in the current class file can call it. | **Note:** *Protected* is another important modifier. You do not need to introduce this. But, if any student asks, the definition is: method/function is accessible within package and outside the package but through inheritance only. |

* 1. Tell the student that every method/function they will write during the camp will be in public.
  2. Type “public” in front of the name of the method/function.
  3. Computer needs to know if it needs to give you back any information. This is also known as return type.
  4. For example, when you use a calculator and type in “546 X 332” — do you want the computer to calculate it and just sit there? Or do you want it to tell you the answer?
  5. Void is used when you do not need the method/function to give any information back. In computer language, “void” means nothing.
  6. Type “void” in after “public” but before the method/function name.

1. Computer needs to know where the instructions starts and ends. We do this with curly brackets ( { } ).
   1. If you forget the brackets, the computer will get confused and doesn’t know when to start or stop reading.

|  |  |
| --- | --- |
| * 1. Write both open curly brackets ( { ) and close curly bracket ( } ) after the the method/function name with some spaces in between to write the programming statements. | **Note:** *BlueJ will automatically create a colored box making it easier for students to identify the block area of the code and where to write.* |

1. Now you should have something like this on the screen:

|  |
| --- |
| public void buildGame(){  } |

1. To build HTL game, we need to add Wizards to the scene. We will have to give the computer a command that it has already known, which are:

|  |  |
| --- | --- |
| * 1. *drawMedicWizard()*: Draws a Medic Wizard at a random tile   2. *drawSpeedyWizard()*: Draws a Speedy Wizard at a random tile | **Note:**   * Medic Wizard * Speedy Wizard |

1. Write drawMedicWizard() and drawSpeedyWizard() within the curly bracket like this:

|  |
| --- |
| public void buildGame(){  drawMedicWizard();  drawSpeedyWizard();  } |

1. Then, compile and run the game. One of each Wizards will show up in the scene.
2. Give students time to explore. Leave your work on the screen for reference.
   1. Challenge students to add 5 or 10 Wizards
   2. Students will likely ask how to change the kind of wizard or put them at specific locations. Tell them they will learn those things later. In the meantime, they can put as many Wizard on the screen as they want.
3. After students have done this, start a discussion.
   1. What are the parts of a method/function?
   2. Can you set up the whole game with just the drawMedicWizard() and drawSpeedyWizard() methods/functions? Why not? What else do we need to do?

# [Lab 2] Method: Drawing Wizards with Parameters

Lab

Concept:

Using parameters to provide information to methods/functions.

Learning Objective:

Student will be able to use a method/function that requires parameters.

How to Assess (optional):

Observe students using methods/functions with parameters in the lab. Ask students probing questions, such as:

1. What is parameters?
2. What parameters are you using? What do they do?

Materials:

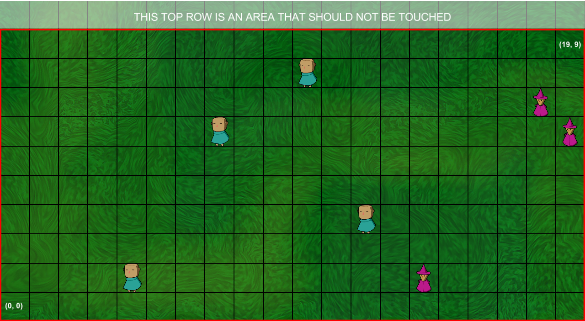
* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer
* Whiteboard and markers (Optional)

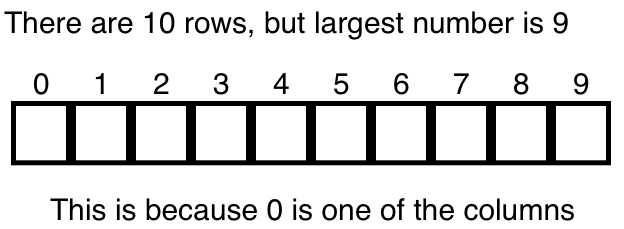
Process:

1. Open BlueJ and go into the “MyGame” class.
2. Recall what the student did from the previous lab: It was writing a method or a function to add Wizards to the game scene by using drawMedicWizard() and drawSpeedyWizard() statements.
3. Start a discussion:
   1. What does the code do?
   2. What DOES NOT it do that you want it to do?
4. Tell the students that they will be learning how to call a method/function with parameters so that they can make the Wizards be at the specific place they want.
5. What is a parameter?
   1. Some methods/functions will require one or more pieces of information (like numbers or strings) in order to complete it tasks.
   2. The pieces of information are known as *Parameters*.
6. Tell the student to think of the game screen as a grid. Each location can be identified using the (x, y) coordinate system. If students do not understand, review it:
   1. Every point start at the bottom left corner, the origin (0, 0).

|  |  |
| --- | --- |
| * 1. x is how many places to move to right from the origin.   2. y is how many places to move up from the origin. | **Note:** Coordinate system used in Java is the 1st quadrant in the Cartesian Coordinate System: the origin is at the bottom left corner.) |

* 1. You may use the figure shown below as the guideline for the game coordinate system.



1. Demonstrate by adding a few Wizards to the scene, for example:
   1. drawMedicWizard(3, 5);
   2. drawSpeedyWizard(10, 9);
2. Give students time to explore. Their task is to draw something interesting using the Wizards. Leave your work on the screen for reference.
   1. Challenge students to figure out the size of the grid: What is the largest x number (should be 19) and y (should be 9) they can put in before the Wizard goes off the screen.
3. After students have done this, start a discussion.
   1. Let students share the numbers they got for the max and min values.
   2. What would happen if they use the x or y number that is bigger than the max and min value?
   3. Why are the min values of both x and y are 0s, but not 1s?
      1. That is because computers often start counting from 0!
      2. For example, if you tell your computer that you want a table consisting of 1 row and 10 columns, the computer will give you this:  
         
         1. You may show this image or draw it on a whiteboard
         2. Pointed out that the first column is counted as 0.
         3. You may also count each column to show that it is really 10 columns.
      3. So, for our game, what do you think is the total number of rows and columns of the game scene (according from the max and min value they found)? They are 10 rows and 20 columns!
4. You may let students share their work to the class.

***Extension (or possibly just a class discussion):***

What if I declared int flower = 3 at the beginning of my program? Could I draw a Wizard by saying:

drawMedicWizard(flower, flower);

or

drawSpeedyWizard(flower, flower);

Try it! Named variables can be used as parameters!

# 

# [Lab 3] If/Else: Draw Wizards with If/Else Statement

Lab

Concept:

Conditionals

Learning Objective:

Student will be able to demonstrate an understanding of conditionals.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Open BlueJ and go into the “MyGame” class.
2. Recall what the student have learned: It was conditionals or if/else statements. They look like this:

|  |
| --- |
| if( *condition* ){  // do something  }  else if( *condition* ){  // do something  }  else{  // do something  } |

1. Inform them that they will be implementing if/else statements in their code today.
2. Tell them that the code you are going to demonstrate today is going to draw the Wizards at the specific place when the arrow key is clicked!
3. Discuss:
   1. Where should this piece of code go? Why?
   2. You may review the concept of “**BuildGame and UpdateGame**” if necessary.
4. Now, begin the demonstration by starting if statement in the updateGame() method, like this:

|  |
| --- |
| public void updateGame(){  if( *something happen* ){  // do something  }  } |

1. Tell them that what you want the game to draw a Wizard when the right arrow key on the keyboard is pressed.
   1. Replace “*something happen*” within the parenthesis to “*keyboardIsPressingRight()*”. Tell the student that this coding statement will tell the computer detect when the right arrow key is pressed.

|  |  |
| --- | --- |
| * 1. Replace “// do something” to drawMedicWizard(20, 5). Tell the student that this coding statement will make the computer draw a Medic Wizard at the position 20 to the right and 5 to the top on the screen. | **Note:** drawMedicWizard(20, 5) can be substituted with drawSpeedyWizard(20, 5) to draw a Speedy Wizard instead of Medic Wizard. |

* 1. Your code should be similar to the code below.

|  |
| --- |
| public void updateGame(){  if( *keyboardIsPressingRight()* ){  drawMedicWizard(20, 5);  }  } |

1. Compile and run the game. Press the right arrow key on the keyboard, then the Medic Wizard will be drawn at the position 20 to the right and 5 to the top.
2. Now, tell the student that you also want to draw another Wizard when the left arrow key on the keyboard is pressed. This can be done by using else if statement.
   1. Follow the same instruction for if statement on step Number 6, but now you will draw a Medic Wizard at position 1 to the right and 5 to the top, drawMedicWizard(1, 5).
   2. You should end up with something on the screen like this:

|  |
| --- |
| public void updateGame(){  if( *keyboardIsPressingRight()* ){  drawMedicWizard(20, 5);  }  else if( *keyboardIsPressingRight()* ){  drawMedicWizard(1, 5);  }  } |

* 1. Make sure the mention that else if statement can only work when there is an if statement existed before it.

1. Compile and run the game. Now, try pressing left arrow key and right arrow key on the keyboard (one key at a time). 2 Wizards should be drawn on the screen.
2. Give students time to explore. Their task is to draw 4 Wizards corresponding to each of the arrow keys being pressed by using if/else statement.
   1. Here are methods/functions for checking if the arrow key is pressed (Make sure that the student use all of them):
      1. keyboardIsPressingLeft()
      2. keyboardIsPressingRight()
      3. keyboardIsPressingUp()
      4. keyboardIsPressingDown()
3. Discussion:
   1. What do if/else statements allow us to do?

# 

# 

# [Lab 4] If/Else: Using Variable Declarations

Concept:

Conditionals and variable declaration

Learning Objective:

Student will be able to demonstrate an understanding of conditionals and variable declarations.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer
* (Optional) Graph paper and pencils for student

Process:

1. Open BlueJ and go into the “MyGame” class.
2. Recall what the student did from the previous lab: It was using if/else statements to draw the Wizard at a specific location when the arrow key on the keyboard is pressed.
3. Tell the student that now we can also draw a new Wizard every time the arrow keys are pressed by using the same idea as the previous lab combining with the concept that they have already learned, variable (You may review the concept of if statement if necessary.).
4. First of all, we will start coding in the buildGame() to create the world for the game.
5. Tell the student to choose a wizard as the main character: either Speedy or Medic Wizard. For the demonstration, the main character is Speedy Wizard.
6. Draw the chosen Wizard at the desired starting point. For the demonstration, the starting point is at the position (0, 0).
7. Tell the student that they will use the other Wizard to build a “maze”.
8. You will use the pre-built maze for this demonstration.
   1. Show them the code and explain how they suppose the build the maze.
      1. Basically, drawing Wizards as a maze wall with the following methods/functions:
         1. drawMedicWizard(int x, int y);
         2. drawSpeedyWizard(int x, int y);
9. You may compile and run the game.
   1. Show the student how the maze should look like.
   2. You may ask:
      1. Where the main character Wizard is shown on the game scene?  
         *Answer:* at the bottom left corner/at the origin/at (0,0)
      2. Why is it at there?
10. Now, tell them that you want to use the arrow keys to draw a line of Wizards through the maze. And, to do that, we will need to use variable.
11. We will need to keep track of the position to draw the main character Wizards.
    1. Review the concept of variable if necessary.
    2. Start by declaring the starting position of the Wizard at the top. For the demonstration, you should declare int x = 0 on the outside of buildGame(); like this:

|  |
| --- |
| int x = 0;  public void buildGame(){  drawSpeedyWizard(0, 0);  …  } |

* 1. In updateGame() function, write an if statement so that if the right arrow key on the keyboard is pressed, then make the main character Wizard move to the right 1 unit. In other words, increase the x value by 1 from what it was before (x = x + 1;).
  2. Now, we need to draw a new main character Wizard at that location:
     1. drawMedicWizard(int x, int y); -- use this function, if the main character is the Medic Wizard.
     2. drawSpeedyWizard(int x, int y); -- use this function, if the main character is the Speedy Wizard.
  3. It should look like this by the end:

|  |
| --- |
| public void updateGame(){  if( *keyboardIsPressingRight()* ){  x = x + 1  drawSpeedyWizard(x, y);  }  } |

1. Give students time to explore. Their task is:
   1. Draw a maze
      1. Allow students to plan out their maze on graph paper, if they like.
      2. They should use at least 10 Wizard to make the maze.
   2. Write if/else statements for pressing right, left, up, down.
      1. Teacher hint: be prepared for a lot of questions about when to add/subtract from x and y
   3. This will take sometimes. Walk around the room to provide them support.
   4. After students have done this, start a discussion.
   5. This is a complicated topic, so you may ask if there are any questions.
      1. You may bring up questions that were asked during the lab time which others might find helpful.
   6. What did they do?
      1. You may let students share their work to the class.
   7. What did they learn from the lab?

# 

# 

# [Lab 5] Nested If: Moving Wizards

# 

Concept:

Nested if statement.

Learning Objective:

Student will be able to demonstrate an understanding of nested if statement.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Open BlueJ and go into the “MyGame” class.
2. Tell them that they will be learning how to use a nested if statement.
3. Explain to them what nested if statement is. (You may review the concept of if statement if necessary.):
   1. Nested if statement is basically having one or more if statements inside of an if statement (also, in else if and else), that look something like this:

|  |
| --- |
| if( *condition* ){  if( *condition* ){  // do something  }  } |

* 1. You can nested as many if statement as needed.

1. Let’s use nested if statements in our code so that we are able to pick up Wizards and move them around.
   1. First of all, we will need to ask the computer if check if the mouse has ever been clicked anywhere on the game scene.
      1. Let them write this piece of code by themselves.
      2. Tell them to use mouseClicked() function. This function will signal the computer whenever the mouse is clicked on the game scene
      3. They should have something like this:

|  |
| --- |
| if( *mouseClicked()* ){  } |

* 1. Now, in the inside, we need to get the position that the mouse that is clicked.
     1. We are going to create 2 variables to store the position x and y of the mouse clicked.
     2. We can get the position of the mouse clicked by using getClickedX() and getClickedY().
     3. And, you should have something like this:

|  |
| --- |
| if( *mouseClicked()* ){  int clickedX = getClickedX();  int clickedY = getClickedY();  } |

* 1. After we get the position of the mouse clicked, we will ask the computer if there is any Wizards at that x and y position.
     1. Let them try write this piece of code by themselves.
     2. Tell them to use tileHasWizard(int x, int y) function. This function will signal the computer if there is a Wizard at the given position x and y.
     3. You may give them a hint that the if statement is pretty straightforward as the first if statement we wrote.

|  |
| --- |
| if( *mouseClicked()* ){  int clickedX = getClickedX();  int clickedY = getClickedY();  if (tileHasWizard(clickedX, clickedY)) {  }  } |

* 1. Now, inside of this if statement, we will check if the Wizard at the mouse clicked position has already been selected or not. It the Wizard has already been selected, unselect it. Otherwise, select it.
     1. We will do this by using the following functions:
        1. wizardIsSelected(int x, int y);
        2. unselectWizard();
        3. selectWizard(int x, int y);
     2. So, we can do

|  |
| --- |
| if( *mouseClicked()* ){  int clickedX = getClickedX();  int clickedY = getClickedY();  if (tileHasWizard(clickedX, clickedY)) {  if (wizardIsSelected(clickedX, clickedY)) {  unselectWizard();  }  else {  selectWizard(clickedX, clickedY);  }  }  } |

* 1. Now, we are able to select and unselect any Wizards in the game scene. However, we are not yet able to move the selected Wizard to a new place.
     1. Let them try write this piece of code by themselves.
     2. We can do this by using the following functions:
        1. aWizardIsSelected();
        2. moveWizardTo(int x, int y);
     3. Hint: this code need to be an else if statement after checking if the there any Wizard selected.
     4. The end result should be similar to this:

|  |
| --- |
| if( *mouseClicked()* ){  int clickedX = getClickedX();  int clickedY = getClickedY();  if (tileHasWizard(clickedX, clickedY)) {  if (wizardIsSelected(clickedX, clickedY)) {  unselectWizard();  }  else {  selectWizard(clickedX, clickedY);  }  }  else if (aWizardIsSelected()) {  moveWizardTo(clickedX, clickedY);  }  } |

* 1. Now, we are successfully able to select, unselect, and move Wizards!
  2. Moreover, we can also make the game more convenient by letting the user draw a new Wizard by using a mouse clicked!
     1. Let them figure this out by themself.
     2. Hint:
        1. It should be in else statement at the end.
        2. They should be using drawSpeedyWizard(int x, int y) or drawMedicWizard(int x, int y) to draw Wizard.
     3. The final result is:

|  |
| --- |
| if( *mouseClicked()* ){  int clickedX = getClickedX();  int clickedY = getClickedY();  if (tileHasWizard(clickedX, clickedY)) {  if (wizardIsSelected(clickedX, clickedY)) {  unselectWizard();  }  else {  selectWizard(clickedX, clickedY);  }  }  else if (aWizardIsSelected()) {  moveWizardTo(clickedX, clickedY);  }  else {  //drawMedicWizard(clickedX, clickedY);  drawSpeedyWizard(clickedX, clickedY);  }  } |

1. Discussion:
   1. How many level of nested if statements we have in this code?  
      *Answer*: 3 levels.
   2. Do you think there is a limit of number of level of nested if statements?

# 

# 

# [Lab 6] If/Else: Personalization

Lab(computer), 90-120 Minutes (depending on group’s comfort with material)

Concept:

Personalized graphic in the game

Learning Objective:

Student will be able to demonstrate an understanding of customizing graphic by replacing files in the resource folder and also concepts covered throughout the week.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer
* (Optional) Graphic editor software that allows drawing and resizing the dimension of pictures, such as MS Paint or Photoshop.

Process:

1. Started the class with a review discussion of the previously learned concept:
   1. Methods/functions
   2. Methods/functions with parameters
   3. If/else statement
   4. Nested If/else statement
2. Tell the students that today they will be learning how to personalized the background of the game and the Wizards.
3. First, have students put pictures on their computer:
   1. They can draw pictures using any drawing software is available.
   2. They can search for pictures online, perhaps from <http://www.school-clip-art.com/> or another age appropriate site.
4. Let student know that the picture will show up in the best quality if the pictures have the following size:
   1. Background -- 1280 x 720
   2. Wizards-- 64 x 64
5. If students could not find the recommended size, demonstrate how to resize the pictures using the graphic editor software or online resizing tool such as <http://www120.lunapic.com/editor/>
6. Show the students where the resources folder is at.
   1. Background: MyGame/resources/art/Background/Path-Background Pieces
   2. Wizards: MyGame/resources/art/Characters
7. They will need to save their picture file with the specific name that the computer recognized/looking for so that they personalized image can be displayed :
   1. **Background:** HTL\_BG\_Grass2.png
   2. **Medic Wizard:** HTL\_Wizard\_Heal\_Idle.png
   3. **Speedy Wizard:** HTL\_Wizard\_Speed\_Idle.png
8. Recompile the code (this should be the code from the lab 4) and run the game.
9. Give students time to explore.
   1. Challenge them to make their own game using their new background and Wizards.

# 

# [Lab 7] For Loop: Create Custom Paths

Lab (computer), 90 -120 Minutes (depending on group’s comfort with material)

Concept:

For loops

Learning Objective:

Student will be able to demonstrate an understanding of the concept by drawing completed custom paths by using for loops.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Open BlueJ and go into the “MyGame” class.
2. Recall what the student have learned: Loop is a repetition structure that call the same code repeatedly. It makes less work for the programmer! It looks something like this:

|  |  |
| --- | --- |
| for (*initialization*; *condition*; *afterthought*){  // do something  } | **Note:** these terms (initialization, condition, and afterthought) should only be used for teacher’s guide line. |

1. Tell the student that today they will be drawing walking paths to the game scene.
2. We will be building the walking path using these methods/functions:

|  |  |
| --- | --- |
| addPathUpDown(int x, int y); | addPathLeftRight(int x, int y); |
| addPathUpLeft(int x, int y) | addPathUpRight(int x, int y); |
| addPathDownLeft(int x, int y); | addPathDownRight(int x, int y); |

1. First, we are going to start coding in the buildGame() method/function.
2. Create a simple straight path. You can use the pre-built path that looks like this:

|  |
| --- |
| addPathLeftRight(0, 5);  addPathLeftRight(1, 5);  ...  addPathLeftRight(18, 5);  addPathLeftRight(19, 5); |

1. You may compile and run the game.
   1. Show the student how the path looks like.
   2. Point out that a simple path may require many lines of code, but, with for loops, we can make the code shorter and simpler.
2. Now, you will replace the pre-built path with the for loop.
   1. *Initialization:*
      1. What is the number we want to start counting from?
      2. In our case, we to start counting from 0. Type this down:

|  |
| --- |
| for( int currentNum = 0; ; ) |

* 1. *Condition:*
     1. What is the condition for the loop to stay running?
     2. In our case, we want the loop to keep running as long as the value of currentNum is less than 20. Add this to the statement:

|  |
| --- |
| for( int currentNum = 0; currentNum < 20; ) |

* 1. *Afterthought:*
     1. What should happen to the every time the loop is running?
     2. In our case, we want to increase the value of currentNum every time the loop is run. Add this to the statement:

|  |
| --- |
| for( int currentNum = 0; currentNum < 20; currentNum = currentNum + 1 ) |

* 1. Add the curly brackets with space to type in.

1. Inside the for loop, put addPathLeftRight(int x, int y);.
   1. *X location (location to the right of origin):*
      1. We want to create paths from the far left, where x = 0, to the far right, where x = 19, of the game scene.
      2. Since the for loop is counting currentNum from 0 to 19 (less than 20), we can use apply currentNum to the parameter x.
   2. *Y location (location to the top of origin):*
      1. We want to draw the paths at y = 5.
   3. So, we should call addPathLeftRight(currentNum, 5); inside the for loop, and it should look something like this:

|  |
| --- |
| for( int currentNum = 0; currentNum < 20; currentNum = currentNum + 1 ){  addPathLeftRight(currentNum, 5);  } |

1. Slowly, walk through the for loop to show the student what it works. (Show them at least until currentNum = 3).
   1. currentNum starts at 0.
   2. The computer is going to check if currentNum is less than 20; IT IS!
   3. Then, add the “LeftRight” path at the position (0, 5).
      1. You may instead ask the class what is the position being drawn at.
   4. Add 1 to the currentNum. not currentNum is 1.
   5. Now, the computer again checks if the currentNum is less than 20; IT IS!
   6. Then, add the “LeftRight” path at the position (1, 5).
   7. Add 1 to the currentNum. not currentNum is 2.
   8. … so no ...
2. You may ask:
   1. When does the loop stop?
      1. When the currentNum is 20.
   2. Why does the loop stop?
      1. currentNum is not less than 20
3. Now, we will add the Walker to the game scene:
   1. First of all, after creating the path, we need to tell Walkers where they should start walking from and should walk to. We can do that by adding a method/function preparePathForWalker(int x, int y, int x, int y) at the end of where the path is created in the code.
      1. The first x and y variables are the x and y positions where the Walker should start walking from
      2. The second x and y variables are the x and y positions where the Walker should walk to.
   2. For our path, it starts at (0, 5) and ends at (19, 5). It should look like this:

|  |
| --- |
| for (int currentNum = 0; currentNum < 20; currentNum = currentNum + 1) {  addPathLeftRight(currentNum, 5);  }  preparePathForWalkers(0,5,19,5); |

|  |  |
| --- | --- |
| * 1. Now, we can add the Walker to the game by adding addBasicWalker() or addQuickWalker() to the updateGame() method/function. However, if we run the game, we will see that there are too many Walkers walking at once.   2. To prevent that, we can use a countdown timer so that our game knows how long to wait before adding a new Walker to the game. For now, we will use the game standard default countdown time, which is 3 seconds. (We don’t need to set anything.) | **Note:**   * Basic Walker * Quick Walker |
| **Note:** if any student wants to set their own countdown time, they can call this function: setCountdown(double seconds) |

* 1. Now, write an if statement that adds a Basic Walker to the scene at every 3 seconds.
     1. We will need to use a countdownFired() method/function, since this method/function checks the countdown timer and signals when the timer reaches 3 seconds.
     2. So, we should have this:

|  |
| --- |
| if ( countdownFired() ){  addBasicWalker();  } |

1. Compile and run the game. The path should look exactly like the one you show students in the beginning of the lab.
2. Give students time to explore. Their task is to create their own paths using methods/functions given in step 4. Their path must involve at least one for loop.
3. Discussion:
   1. Do you think using loop is better than not using loop? Why?
   2. Does loop make things easier for programmer?

# 

# [Lab 8] While Loop: What Is While Loop?

Lab (computer), 90 -120 Minutes (depending on group’s comfort with material)

Concept:

While loops

Learning Objective:

Student will be able to demonstrate an understanding of:

* The concept by drawing completed custom paths by using while loops
* The different and the similarity between for loop and while loop.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Open BlueJ and go into the “MyGame” class.
2. Started the class with a review discussion of the concept learned in previous lab:
   1. What did we do in the previous lab?
      1. using for loops to draw the walking paths on the game scene.
   2. What is the for loop format?

|  |
| --- |
| for (*initialization*; *condition*; *afterthought*){  // do something  } |

1. Tell them that they are going to learn another type of loop, called while loop.
2. What is while loop?
   1. A kind of loop that only requires *Condition* in order for the loop to run.
   2. It looks like this:

|  |
| --- |
| while ( condition ){  // do something  } |

* 1. It can do everything that the for loop can.

1. So, let’s rewrite the for loop in the previous lab to the while loop!
   1. Replace for loop for while loop structure complete. You should have this:

|  |
| --- |
| while ( ){  } |

* 1. Remember, in for loop, we have to create the count variable to tell the computer “what is the number we want to start counting from.” We need to do the same thing here. But, we need to create the count variable outside of the loop this time because the while loop structure only does not require this piece of information. Type this:

|  |
| --- |
| int currentNum = 0;  while ( ){  } |

* 1. Now, we need to add the *Condition* to the loop to tell the computer “what is the condition for the loop to stay running.” This is similar to what we did in the previous lab: we want the loop to run *while* the currentNum is less than 20. We should have:

|  |
| --- |
| int currentNum = 0;  while (currentNum < 20) {  } |

* 1. Similarly to the for loop, we also need tell the computer “what should happen to the every time the loop is running.”: We want to update the variable currentNum by 1 unit every time the loop is run. Add the afterthought inside the while loop like this:

|  |
| --- |
| int currentNum = 0;  while (currentNum < 20) {  currentNum = currentNum + 1;  } |

* 1. Lastly, add addPathLeftRight() to the loop so that the path will be drawn. The method/function should be added before the last line.

|  |
| --- |
| int currentNum = 0;  while (currentNum < 20) {  addPathLeftRight(currentNum, 5);  currentNum = currentNum + 1;  } |

1. Don’t forgot to prepare path for Walkers and add them to the game! It should be just like what we did in Lab 7.
2. Compile and run the game. The path should look exactly like the one from the previous lab.
3. Give students time to explore. Their task is to create their own paths involving the use of while loop.
4. Discuss:
   1. What is the difference and similarly between while loop and for loop?
   2. Which one, between for loop and while loop, is easier? (or Which one they like better?)
   3. What would happen if you update the variable currentNum before calling the function/method addPathLeftRight().
      1. *Answer*:The path will get shift 1 unit, it will start at 1 and end at 20 instead of start at 0 and end at 19.

# 

# 

# [Lab 9] Nested Loop: Make Wizards Casting Spells

# 

Concept:

Nested for loop

Learning Objective:

Student will be able to demonstrate an understanding of the concept of by using nested for loop to make Wizards casting spells on Walkers if there are nearby.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Open BlueJ and go into the “MyGame” class.
2. Started the class with a review discussion of the all loop concepts they have previously learned:
   1. How many loop have we learned so far?
   2. You may review the concept of for loop and while loop if necessary.
3. Tell the student that today they will be learning on how to use the nested loop in this lab.
4. Explain or ask them what (they think) nested loop is:
   1. Nested loop is pretty similar to nested if statements: having one or more loops inside of a loop.
   2. For example, this is a nested for loop:

|  |
| --- |
| for (*initialization*; *condition*; *afterthought*){  for (*initialization*; *condition*; *afterthought*){  // do something  }  } |

* 1. Nested loop can be in any form:
     + Nested for loop: for loop inside of for loop
     + Nested while loop: while loop inside of while loop
     + While loop nested in for loop
     + For loop nested in while loop

1. Let’s think of what we did last week, creating a straight path using a loop!
   1. Discuss: What is the limitation?  
      *Answer*: You can only draw one straight line using a loop.
   2. With the power of nested loop, we will be able use just 2 loops to fill the whole screen with Wizards!
2. To fill the whole screen with Wizards by using only 2 for loop, we need to do the following:
   1. First, we are going to start by drawing a 3x3 block of Wizards at the bottom left corner by hand. You should have something like this:

|  |
| --- |
| /\* Column 0 \*/  drawMedicWizard(0, 0); drawMedicWizard(0, 1);  drawMedicWizard(0, 2);  /\* Column 1 \*/ drawMedicWizard(1, 0);  drawMedicWizard(1, 1);  drawMedicWizard(1, 2);  /\* Column 2 \*/  drawMedicWizard(2, 0);  drawMedicWizard(2, 1);  drawMedicWizard(2, 2); |

* 1. Point out for the student to see that the x values of each column is simply just counting up by one value per section, we can use for loop to represent the same thing, like this:

|  |
| --- |
| for(int i = 0; i < 2; i = i + 1){  drawMedicWizard(i, 0);  drawMedicWizard(i, 1);  drawMedicWizard(i, 2);  } |

* 1. You may trace through the code to show the student that this piece of code is doing exactly what the one before does.
  2. Now, ask the student if they notice anything (inside the for loop) that can also be turned into another for loop? If they couldn’t, point out the y values.
  3. Similarly to the x value, the y value is just simply counting up by a value as well. So, we can also turn those into another for loop exactly just like how we did with the x value. So, the final result should look like this:

|  |
| --- |
| for(int i = 0; i <= 2; i = i + 1){  for(int j = 0; j <= 2; j = j + 1){  drawMedicWizard(i, j);  }  } |

* + - Again, emphasize the outer for loop is taking care of counting for the x values, while the inner for loop is taking care of counting for the y value
  1. Now, tell them to try filling the whole screen with Wizards using nested for loop. This should not take long, give them 5 - 10 minutes to work on this and asking questions.

1. Now, we are going to use nested for loop to make the Wizard casting the spell on the Walker that walks pass.
2. First of all, we need to have path, Walkers, and Wizards ready before doing this lab. So, for now, we will take the code from our previous lab, Lab 8 and Lab 5.
3. Before we start coding, we have to understand how we are going to do this:
   1. The computer keeps track of every Wizards and Walkers added to the game scene in a line of chronological order. It looks like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | ... |

*All Wizards added to the game*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | ... |

*All Walker added to the game*

* 1. Discuss:
     + When will Wizards should cast the spell?
     + How the Wizards know which Walker to cast the spell on?  
       *Answer*: The Wizard needs to check if there are any Walkers are walking nearby
     + What are some ideas for the coding strategy with students?
       1. What we are going to do is to use nested for loop:
          1. The first for loop will go through all Wizards
          2. The second for loop that is nested will go through all Walkers

1. Now, we can start coding!
   1. Tell them they can use the following methods/functions:
      * numOfWizards() -- give total number of Wizards in the game
      * numOfWalkers() -- give total number of Walkers in the game
      * walkerIsInRange(int wizardNumber, int walkerNumber)
      * medicWizaredCastSpellOnWalker(int wizardNumber, int walkerNumber)
      * speedyWizardCastSpellOnWalker(int wizardNumber, int walkerNumber)
   2. Give them time to implement this by himself, and play each other’s games once they are done.
   3. The final result should be similar to this:

|  |
| --- |
| for (int i = 0; i < numOfWizards(); i = i + 1){  for(int j = 0; j < numOfWalkers(); j = j + 1){  if(walkerIsInRange(i, j){  //medicWizardCastSpellOnWalker(i, j);  speedyWizardCastSpellOnWalker(i, j);  }  }  } |

1. When most of the class seems to have finished, demonstrate the solution on the projector and go over it line by line.
2. Discuss:
   1. What do you think

# 

# 

# [Lab 10] Gameplay: Win Conditions

Concept:

Personalized graphic in the game

Learning Objective:

Student will be able to demonstrate an understanding of all concepts learned from the previous labs.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Open BlueJ and go into the “MyGame” class.
2. So, first, let’s think of some of your favorite games.
   1. How do you win?
   2. How do you know you’ve won?
3. Now, think of our game
   1. How do you win? (Teacher hint: We haven’t made a way to win yet!)
   2. What are some ways we could make it so that you could win?
   3. What should be the win conditions? (Teacher hint: Try to steer the conversation towards counting number of Walker and their health points. It’s ok if they want to add and time those values together.)
4. Frist of all, we need to have path, Walkers, and Wizards ready before doing this lab. So, for now, we will take the code from our previous lab, Lab 8.
   1. Also, add drawToolbars() method/function to the top of buildGame() to add user interface that shows score information of the game.
5. we need to have a way to count the game score.
   1. Tell them they can use the following methods/functions:
      1. updateScore(float score)
      2. getNumOfWalkersSaved()
      3. getHealthSaved()
   2. Give them time to implement and experiment with the method/function.
   3. Teacher Hint: This should be place somewhere at the bottom of the updateGame().
6. Now, let’s write some code!
   1. Tell them they will be writing a new method/function today!
   2. Instruct them to set up their methods/functions the same way the set up buildGame() method/function with “public” and “void”. The different is that we will name this method/function checkGameWon().
   3. Inside the method/function, write the coding statement that check the game current score yet pass the winning condition. If it pass, displaying the winning screen.
      1. Tell them to use the following methods/functions
         1. use getScore()
         2. enterWin()
      2. They can test these methods/functions within the updateGame() method/function. Depending on how confident they are, you may want to demonstrate calling this method for them on the projector.
      3. Give them time to implement this by himself, and play each other’s games once they are done.
      4. The final result should be similar to this:

|  |
| --- |
| if( getScore() >= 500 ){  enterWin();  } |

1. Discuss:
   1. What is fun and not fun?
   2. If you were making this game for younger kids, would you change the number of winning condition? What about college students?
   3. Is it important to consider who will be playing the game?

# 

# 

# [Lab 11] Gameplay: Cleanup The Game!

Concept:

Personalized graphic in the game

Learning Objective:

Student will be able to demonstrate an understanding of all concepts learned from the previous labs.

How to Assess (optional):

Observe students work on their computer

Materials:

* A projector showing the teacher’s screen
* A computer with BlueJ for each student
* The HTLFunctionalAPI installed on each computer

Process:

1. Open BlueJ and go into the “MyGame” class.
2. Started the class with a review discussion about our game.
   1. How the game is like?
   2. What is the game limitation? (Teacher hint: Try to steer the conversation towards adding one type of Wizards and Walkers at the time.)
3. Let’s add a new gameplay element to the game so that we can add any kind of Wizards and Walkers in the game. But, before that let’s discuss:
   1. Does this feature make the game more intergesting? more fun? harder?
   2. Let’s brainstorm on what techniques and coding concepts we can use?
      1. Emphasize that this would involve the use of calling methods/functions, if statement, and variables.
      2. Teacher hint: This should be similar to Lab4.
4. Now, let’s write some code!
   1. a. Tell them they can use the following methods/functions:
      1. keyboardIsPressingLeft()
      2. keyboardIsPressingRight()
      3. keyboardIsPressingUp()
      4. keyboardIsPressingUp()
      5. addWalker(String type): *basic* and *quick*
      6. drawWizard(int x, int y, String type): *medic* and *speedy*
   2. Inform the class that addWalker(String type) and drawWizard(int x, int y, String type) just need to be replaced the old ones they had in the Lab10. Those old methods/functions are: addBasicWalker(), addQuickWalker(), drawSpeedyWizard(int x, int y), or drawMedicWizard(int x, int y).
   3. Give them time to experiment.
   4. If you see that they are struggling, provide them hints such as:
      1. Tell them to take a look at Lab4.
      2. Demonstrate one if statement as an example.

|  |
| --- |
| if (keyboardIsPressingLeft()) {  defaultWizardType = "medic";  } |

1. When most of the class seems to have finished, demonstrate the solution on the projector (reinforces concept).
   1. Discuss: where should these codes (if statments) be?
      1. Let them brainstorm and guess.
      2. *Answer:* the code should be anywhere but before drawWizard(int x, int y, String type) or addWalker(String type), so the variable information is updated before passing it to the method/function!
2. Now, notice anything about the gameplay? Are there anything you wish to modify to make the game more fun and interesting?
   1. The methods/functions they can use are:
      1. setCountdown(double seconds)
      2. setWalkerDamagePerSecond(int damage)
      3. setSpeedyTimeBetweenSpellcasts(double duration)
      4. setMedicTimeBetweenSpellcasts(double duration)
      5. setMedicWizardHealthAdjust(double health)
      6. setSpeedyWizardSpeedBoostDuration(double duration)
      7. setSpeedyWizardSpeedBoostMultipler(double multiplier)
   2. Give the time to play with these new methods and create a fun game.
3. Discuss:
   1. What settings did you come up with that were fun?
   2. Which methods/functions can take negative values? What is the difference when you use negative values?
   3. Who did you make this game for (what audience)?