# TINKER ACADEMY

## Programming Using Java

**Handout 4: Classes, Objects and Methods**

Note your Student ID. You will need to use it throughout the Course.

### Setup Instructions In Classroom

Connect to the Local Class Network

1. Select WiFi “TINKER ACADEMY”
2. This network has only LOCAL access and does NOT connect to the internet

Update the Course

1. Ensure you are connected to “TINKER ACADEMY”
2. Restart the VM. Login into the VM.
3. Open Firefox in the VM
4. Your Instructor would tell you what to type in the browser. (Typically it is 192.168.1.5)
5. You should see a page with a list of entries.
6. Click on CourseUpdate<Date>.zip. This will download CourseUpdate<Date>.zip onto your VM
7. Open Nautilus. Click on Downloads. You should see the file CourseUpdate<Date>.zip
8. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
9. Open the extracted folder
10. Double click Course Update. Select “Run” in the window.

Update the Course (Alternate Approach In Class Using USB)

1. Borrow a USB drive from the Instructor
2. If you are on VirtualBox
    a. Click on Devices in the Top level Menu
    b. Select Drag ‘n’ Drop
    c. Select Bidirectional
3. If you are on VirtualBox (Another Way)
    a. Shutdown Virtual Machine
    b. Click on VM in the VirtualBox Manager
    c. Click on the Settings
    d. Click General

    e. Click Advanced Tab
    f. Select “Bidirectional” under Drag ‘n’ Drop
    g. Click OK
    h. Start Virtual Machine
4. If you are on VMWare
    a. Open the virtual machine settings editor (VM > Settings),
    b. Click the Options tab
    c. Select Guest isolation.
    d. Deselect Disable drag and drop to and from this virtual machine
5. Open Nautilus, Click on Desktop
6. Drag the file CourseUpdate<Date>.zip from Windows or Mac onto Desktop in your Virtual Machine
7. Right Click on CourseUpdate<Date>.zip. Select Extract Here.
8. Open the extracted folder
9. Double click Course Update. Select “Run” in the window.
10. Eject the USB Drive and hand it back to the Tinker Academy instructor

## Setup Instructions At Home

Connect to your Home WiFi Network

Updating the Course (Using Wifi)

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Update”. Choose “Run”.
   If you see a window popup with the message “update course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.
4. Follow the instructions in this handout (last 2 pages) on the quiz and homework steps.

Submitting Quiz and Homework

1. Make sure you are on the Home WiFi Network.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Submit”. Choose “Run”.
   If you see a window popup with the message “submit course failed”.
   Hop onto Skype, and request help in the class chat group.
   And send an email to classes@tinkeracademy.com with your name and student ID.

## Virtual Machine Installation

Installing the Virtual Machine (VM)

1. Borrow the USB drive from your Tinker Academy instructor
2. Create the folder “tinkeracademy” (without the quotes) under Documents using Finder or Windows Explorer. Type it in *exactly* as indicated.
3. Copy the folder “installers” from the USB drive to under “tinkeracademy” using Finder or Windows Explorer
4. Eject the USB Drive and hand it back to the Tinker Academy instructor
5. Locate the VirtualBox installer under “tinkeracademy” using Finder or Windows Explorer

| If your Laptop is | Double click on |
|---|---|
| Windows 7 | VirtualBox-4.3.12-93733-Win.exe |
| Windows 8 | VirtualBox-4.3.14-95030-Win.exe |
| Mac | VirtualBox-4.2.26-95022-OSX.dmg |

6. Install the VirtualBox application
7. Congratulations, You completed a major milestone. Give yourself a pat on the back :)

Importing the Virtual Machine (VM)

1. Locate the Virtual Machine “tinkeracademy.ova” under “tinkeracademy”
2. Double click on “tinkeracademy.ova”. You should get the import screen in VirtualBox with an “Import” Button. Click on the “Import” button to Import the Virtual Machine.

Starting the Virtual Machine (VM)

1. Once the Import is complete and successful, you should see the VM “TinkerAcademy” in the side panel in VirtualBox.
2. If it says “Powered Off” click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
3. If it says “Running” click on the Show Button (Green Arrow) in the VirtualBox Toolbar. This should display the VM window.
4. Once the VM starts up you will be presented with a login screen. Type in “password” without the quotes. Type it in exactly as indicated and hit “Enter”.
5. Once the login is completed you should see a Desktop with a few icons. The Screen might go fuzzy for a few seconds before displaying the Desktop. *That is ok.*
6. Congratulations. You are now running Linux within your laptop.
7. Double click on the “Firefox” icon in the Sidebar. This should launch Firefox. Verify you have network access. Close “Firefox”

Launching the Virtual Machine in Full Screen

1. Use the VirtualBox menu View->Switch to Fullscreen to switch the VM to fullscreen mode
2. Use the same VirtualBox menu View->Switch to Fullscreen to switch the VM back out of fullscreen mode

Shutting Down the Virtual Machine

1. Click on the red close window button (to the top left on a Mac, top right in Windows).
2. You will prompted with a confirmation message asking if you want to “Power Off” the machine. Click the button to confirm power off.
3. In a few minutes the VM will shut down and you should see the VirtualBox side panel with the “Tinker academy” VM indicating “Powered Off”.

Restarting the Virtual Machine

1. Start VirtualBox
2. Click on the VM “TinkerAcademy” in the VirtualBox side panel.
3. Click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
4. Once the VM startup you will be presented with a login screen.

Right Click in VM on Mac

1. Open System Preferences, Trackpad
2. Enable “Secondary Click”, Toggle the small arrow to the right and select “Click with two fingers”.

## Getting Ready to Program

**Import StarterPack4.zip**
We will be using StarterPack4.zip for this class.

Click on StarterPack4.zip under “Courses”. Right Click. Select “Extract Here”
Start Eclipse from Desktop. Right Click over Package explorer. Select “Import”. Browse and select the extracted folder “StarterPack”. Click “Finish” to complete the Import.

**Structure of StaterPack4.zip**
StarterPack4 is an Eclipse Java Project. The Project has

1. A Java Source File called “StarterPack4.java”
2. A Java Source File called “MyMinecraftHouse.java”
3. Project references to the Java Runtime Environment 1.6 System Library.
4. Project references to the Minecraft Plugin Library.

**Run the Program**

1. Click over project name “StarterPack4”.
2. Toggle the src folder to select StarterPack4.java
3. Right Click. Select “Run As…”. Select “Java Application”.
   The text “Starter Pack 4” should be displayed in the Console window (bottom)

If the “Console” window is not visible click on the “Window” Toolbar (top), select “Show View”, select “Console”. If all else fails, click on the “Window” Toolbar (top), select “Reset Perspective…” click “OK”. This will put Eclipse back into the default Factory state. You can now use “Show View” to view the “Console”.

## Fundamentals of a Java Class

What is a **class**?
The concept of a class is the foundation on which the entire Java language is built.

Think of it as a blueprint that is used to create **objects** of that type.

What is a **object**?
The object is anything that you can identify.

Every object in Java needs to be based on **one or more blueprints**. Each blueprint represents a class.

Every object interacts with the outside world through **interfaces**.

What is a **interface**?
Objects interact with the outside world through interfaces.

An interface is the list of all methods through which it can interact with the outside world.

Every object in java has one or more interfaces.

The code below declares the variable myHouse of datatype MyMinecraftHouse and assigns a new object of type MyMinecraftHouse as the initial value.

```
MyMinecraftHouse myHouse = new MyMinecraftHouse();
```

**Visual Model**
The visual model below below shows the blueprint for MyMinecraftHouse and the object myHouse based on the blueprint.

| Class | Method 0 | Method 1 | Method 2 | Method 3 |
|---|---|---|---|---|
| MyMinecraftHouse | **Constructor** | | | |

| Object | Blueprint 1 | | | |
|---|---|---|---|---|
| myHouse | MyMinecraftHouse | | | |

## Methods

**What is a method?**

A method is a reusable piece of code that takes some inputs, does some work and (usually) returns some output. A method is defined in a Java Class.
Once a method is defined in a class, other parts of the class and even other classes can call it.
Methods are incredibly useful in programming.

In addition to the methods we had defined in the previous class, the Java Compiler added a special “method” for you called the constructor.

## Constructor

**What is a constructor?**
A constructor is a special method that prepares the new object for use.

Constructors are optional in Java.

The Java Compiler will automagically create the “default” constructor for the class if it does nc have one defined.

Even though constructors look like Java methods, they

- can never be invoked directly*
- do not return anything i.e. have void as the return type
- cannot be made static

* except through the constructor of a child class (We will cover hierarchies in next class)

## Defining a Method

The code below defines the method build for the class MyMinecraftHouse. The method does not require any inputs and does not return any output.

1. Open Eclipse
2. Click project name “StarterPack4” in Package Explorer
3. Open MyMinecraftHouse.java under “src”.
4. Add the code under //TODO (1)

```
public void build() {
```

```
}
```

5. Save the File
6. Open Problems View. You should see no Errors.
7. Double click on the method name build. The entire word should be selected.
8. Right click, Select Source, Select Generate Element Comment. This will generate a new comment for the method.
9. Update the comment similar to that below

```
/**
 * Builds a House
 */
```

10. Save the File
11. Open Problems View. You should see no Errors.

**Visual Model**

The visual model below below shows the blueprint for MyMinecraftHouse and the object myMinecraftHouse based on the blueprint.

| Class | Method 0 | Method 1 | Method 2 | Method 3 |
|---|---|---|---|---|
| MyMinecraftHouse | Constructor | **build** | | |

| Object | Blueprint 1 | | | |
|---|---|---|---|---|
| myHouse | MyMinecraftHouse | | | |

We will later add code to this method to build a house in Minecraft.

## Fields

So far the blueprint for MyMinecraftHouse has only methods defined in it.
A class can also define variables defined as part of its blueprint.

These variables are called fields.

Think of a field as a property that every object of that class will have.

**What is a field?**
A field is a variable defined in a class.

A field has a name and a datatype similar to a variable.
In addition to the name and datatype, a field has modifiers

The access modifier declare
- who has access to the field, i.e who can read the value of the field or assign a new value to the field.

The static modifier declare
- the field is accessed before an object of the class is created
- the field is shared by all objects of the class

The code below defines a new field for the class MyJavaClass1.

The name of the field is name.
The datatype of the field is String

The field will have public access.
This means that other parts of the program will be able to read the value and also assign a new value.

The field does not have a static modifier.
This means that every object of that class will have its own value

1. Open Eclipse
2. Click project name “StarterPack4” in Package Explorer
3. Open MyMinecraftHouse.java under “src”.
4. Add the code under //TODO (2)

```
public String name;
```

12. Save the File
13. Open Problems View. You should see no Errors.

The code below shows the blueprint for MyJavaClass1 and the object myJavaClass1 based on the blueprint after the field name has been added.

| Class | Method 0 | Method 1 | Method 2 | Method 3 |
|---|---|---|---|---|
| MyMinecraftHouse | Constructor | build | | |
| | Field 1 | | | |
| | **name** | | | |

| Object | Blueprint 1 | name | | |
|---|---|---|---|---|
| myHouse | MyMinecraftHouse | null | | |

## Accessing Fields

The field name has public access. This means we can read the value of the field or assign a new value to the field from any other part of the program.

Since this field does not have a static modifier, it means that each object of the class will have its own copy of the field. It also means that an object has to be created before accessing the field.

The value of a field can be read using using the dot (.) operator.

The code below reads the value of the field name for the object myHouse which is an object of class MyMinecraftHouse.

```
String houseName = myHouse.name;
```

The code below accesses the name field of the class MyMinecraftHouse from the main method of StarterPack4.

1. Open Eclipse
2. Click project name “StarterPack4” in Package Explorer
3. Open StarterPack4.java under “src”.
4. Add the code under  //TODO (3)

```
MyMinecraftHouse myHouse = new MyMinecraftHouse();
String houseName = myHouse.name;
System.out.println("My Minecraft House is called " + houseName);
```

14. Save the File
15. Open Problems View. You should see no Errors.
16. Run the program
17. The Console window should display the following

```
Starter Pack 4
My Minecraft House is called null
```

The name of myHouse is null? Seriously?

## Initial value of Fields

The name of myHouse is called null

Remember that null is a special literal that indicates that the variable does not have a value

Every field is initialized to a "default value" based on the datatype of the field.

The default value of the field depends on the datatype.

| Field Datatype | Default Value |
|---|---|
| Any Java Object | null |
| boolean | false |
| int | 0 |
| long | 0 |
| short | 0 |
| byte | 0 |
| float | 0.0 |
| double | 0.0 |

| char | **the 0 character** |
|---|---|

**What? the 0 character!!**

Any character

a, b, c, d ...
0, 1, 2, 3 …

is represented internally as a number. This mapping of a character to a number is called the ASCII code.

Here is a list of mappings from the commonly used characters to their ASCII codes.

Upper Case Alphabets

| ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘H’ | ‘I’ | ‘J’ | ‘K’ | ‘L’ | ‘M’ | ‘N’ | ‘O’ | ‘P’ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| ‘Q’ | ‘R’ | ‘S’ | ‘T’ | ‘U’ | ‘V’ | ‘W’ | ‘X’ | ‘Y’ | ‘Z’ | | | | | | |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | | | | | | |

Lower Case Alphabets

| ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | ‘k’ | ‘l’ | ‘m’ | ‘n’ | ‘o’ | ‘p’ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 |
| ‘q’ | ‘r’ | ‘s’ | ‘t’ | ‘u’ | ‘v’ | ‘w’ | ‘x’ | ‘y’ | ‘z’ | | | | | | |
| 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | | | | | | |

Digits

| ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’ |
|---|---|---|---|---|---|---|---|---|---|
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |

Punctuation Characters

| ‘ ’ | ‘!’ | ‘“’ | ‘#’ | ‘$’ | ‘%’ | ‘&’ | ‘’’ | ‘(’ | ‘)’ | ‘*’ | ‘+’ | ‘,’ | ‘-’ | ‘.’ | ‘/’ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| ‘:’ | ‘;’ | ‘<’ | ‘=’ | ‘>’ | ‘?’ | ‘@’ | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 58 | 59 | 60 | 61 | 62 | 63 | 64 | | | | | | | | | |

Special Characters

| Character | Description | ASCII Code |
|---|---|---|
| '\0' | the 0 character<br><br>(also called the null character) | 0 |
| '\b' | Backspace character | |
| '\t' | Tab character | |
| '\n' | New Line character | |
| '\r' | Carriage Return Character | |

## Modifying Fields

The value of a field can be set using using the dot (.) operator.

The code below set the value of the field name for the object myHouse which is an object of class MyMinecraftHouse.

A new value can be assigned to the field using the dot (.) operator.

```
myHouse.name = "Casa Rosa";
```

The code below modifies the name field of the class MyMinecraftHouse from the main method of StarterPack4.

1. Open Eclipse
2. Click project name "StarterPack4" in Package Explorer
3. Open StarterPack4.java under "src".
4. Add the code under //TODO (4)

```
myHouse.name = "Casa Rosa";
System.out.println("My Minecraft House is now called " + myHouse.name);
```

5. Save the File
6. Open Problems View. You should see no Errors.
7. Run the program
8. The Console window should display the following

```
Starter Pack 4
My Minecraft House is called null
My Minecraft House is now called Casa Rosa
```

## Controlling Access to the Field

The field name has public access, which means any other part of the program can read its value and modify its value.

What if the name of the house should not be modifiable once myHouse is first created.

This is done in Java by changing the access modifier on the field.

The following code changes the access modifier from public to private.

The private modifier indicates that only the object myHouse can modify or read the field directly.

```
private String name;
```

The code below changes the access modifier of the name field of the class MyMinecraftHouse.

1. Open Eclipse
2. Click project name “StarterPack4” in Package Explorer
3. Open MyMinecraftHouse.java under “src”.
4. Modify the code under  //TODO (1)

```
~~public~~ private String name;
```

5. Save the File
6. Open Problems View. You will see a bunch of errors!

You should see errors in the Problems View. That is ok.

The errors show up because the name field had public access but now has private access. This means that no other part of the program will be able to read or modify the value.

We want other parts of the program to read the value but not modify the value.

This can be be done by adding a method that returns the name of the house.

1. Open Eclipse
2. Click project name “StarterPack4” in Package Explorer
3. Open MyMinecraftHouse.java under “src”.
4. Modify the code under //TODO (5)

```
public String getName() {
        return this.name;
}
```

Modify the code under //TODO (3)

```
MyMinecraftHouse myHouse = new MyMinecraftHouse();
String houseName = myHouse.name myHouse.getName();
System.out.println(“My Minecraft House is called ” + houseName);
```

Comment out the code under //TODO (4)

```
// myHouse.name = “Casa Rosa”;
// System.out.println(“My Minecraft House is now called ” + myHouse.name);
```

5. Save the File
7. Open Problems View. You will see no Errors!
8. Run the program
9. The Console window should display the following

```
Starter Pack 4
My Minecraft House is called null
```

null? Again?

There needs to be a way assign the correct name.

## Using the Constructor

**What is a constructor?**
A constructor is a special method that prepares the new object for use.

Constructors are optional in Java.

The Java Compiler will automagically create the "default" constructor for the class if it does nc have one defined.

The purpose of the constructor is to prepare an object for use.

The code below sets the correct name for myHouse.

```
public MyMinecraftHouse() {
        this.name = "Casa Rosa";
}
```

**What is this?**
**this** is a special keyword that refers to the object that is being created.

The access modifier of the field name is private. This means that only the object can access the field. Since **this** refers to the object, it has access to the field.

1. Open Eclipse
2. Click project name "StarterPack4" in Package Explorer
3. Open MyMinecraftHouse.java under "src".
4. Modify the code under //TODO (6)

```
public MyMinecraftHouse() {
        this.name = "Casa Rosa";
}
```

5. Save the File
7. Open Problems View. You will see no Errors!
8. Run the program
9. The Console window should display the following

```
Starter Pack 4
```

```
My Minecraft House is called Casa Rosa
```

Much Better!

But is it good enough?

## Object Creator and Object State

Access control is now in place for the name field.
Only the object can change its value.

However, every object of class MyMinecraftHouse will now name set to “Casa Rosa”.

We want the **object creator** to have control of the initial **object state**.

**What is Object State?**
The state of the object is the set of values of ALL the fields at some point in time.

This is similar to the state of the variable which is the value of the variable at some point in time.

The initial object state is the state of the object when its created.

**Who is the Object creator?**
The Object creator is the part of the program that creates the object (using the new operator).

The object creator needs to be able to set the initial object state.

The class MyMinecraftHouse does not allow the object creator to set the initial object state since all houses will have the same name when they are created.

The following code modifies the constructor to accept the name as an input. The input is of datatype String.

```
public MyMinecraftHouse(String name) {
        this.name = name;
}
```

1. Open Eclipse
2. Click project name “StarterPack4” in Package Explorer
3. Open MyMinecraftHouse.java under “src”.
4. Modify the code under  //TODO (6)

```
public MyMinecraftHouse(String name) {
        this.name = name;
}
```

5. Modify the code under //TODO (3)

```
MyMinecraftHouse myHouse = new MyMinecraftHouse() new MyMinecraftHouse(“Cas
Rosa”);
String houseName = myHouse.getName();
System.out.println(“My Minecraft House is called ” + houseName);
MyMinecraftHouse myOtherHouseName = new MyMinecraftHouse(“Casa Grande”);
String otherHouseName = myOtherHouseName.name;
System.out.println(“My Other Minecraft House is called “ + otherHouseName)
```

6. Save the File
10. Open Problems View. You will see no Errors!
11. Run the program
12. The Console window should display the following

```
Starter Pack 4
My Minecraft House is called Casa Rosa
My Other Minecraft House is called Casa Grande
```

## Adding Fields

The code below adds the width and the height fields to the MyMinecraftHouse class.

The access modifier for both fields is set to **private**.

The initial value of the fields is set through the constructor.

Other parts of the program can access the width and height but cannot change the value by invoking the getWidth and the getHeight methods.

1. Open Eclipse
2. Click project name “StarterPack4” in Package Explorer
3. Open MyMinecraftHouse.java under “src”.
4. Modify the code under  //TODO (6)

```
public MyMinecraftHouse(String name, int width, int height) {
        this.name = name;
        this.width = width;
        this.height = height;
}
```

5. Modify the code under //TODO (7)

```
private int width;

private int height;
public int getWidth() {
        return this.width;
}

public int getHeight() {
        return this.height
}
```

6. Modify the code under //TODO (3)

```
MyMinecraftHouse myHouse = new MyMinecraftHouse(“Casa Rosa”) new
MyMinecraftHouse(“Casa Rosa”, 10, 10);
String houseName = myHouse.getName();
System.out.println(“My Minecraft House is called ” + houseName);
MyMinecraftHouse myOtherHouseName = new MyMinecraftHouse(“Casa Grande”)
new MyMinecraftHouse(“Casa Grande”, 20, 20);
String otherHouseName = myOtherHouseName.name;
System.out.println(“My Other Minecraft House is called “ + otherHouseName)
```

The visual model for the class and object are as follows.

| Class | Method 0 | Method 1 | Method 2 | Method 3 |
|---|---|---|---|---|

| MyMinecraftHouse | Constructor | build | | |
|---|---|---|---|---|
| | Field 1 | Field 2 | Field 3 | |
| | name | **width** | **height** | |

| Object | Blueprint 1 | name | width | height |
|---|---|---|---|---|
| myHouse | MyMinecraftHouse | Casa Rosa | 10 | 10 |

Great! Now we have the name, width and height of the house as part of the object state.

We will implement the build method as part of the House Plugin.

That was a LOT we covered!

You made it this far! Awesome!

We will cover **Interfaces, Class Inheritance and Packages** in the next class class. For now you need to make sure you have a good conceptual understand of **Classes, Objects and Methods**.

**Quiz 4: Classes, Objects and Methods**

Make sure you read this Handout!

## Open the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Updating the Course” in this Handout.
Open Quiz4.odt under “Courses” “TA-JAV-1” “quiz” “quiz4”

## Complete the Quiz

1. Attempt each question. Type in the answers in the “Answer:” box.
2. Save the file using File->Save or Ctrl-S

## Submit the Quiz

Make sure you are on the Home WiFi.
Follow the instructions in “Submitting Homework” in this Handout.

**Homework 4: Classes, Objects and Methods**

Make sure you read this Handout!

## Overview

In this Homework you will define fields and methods to a class called MyMinecraftPlayerInfo (which keeps track of the state of the player). You will add a constructor to initialize the object state.

## Open the Homework

Follow the instructions in “Updating the Course” in this Handout.
Extract Homework4.zip under “Courses” “TA-JAV-1” “homework” “homework4”

- Select Homework zip file
- Right Click, Select Extract Here

Import the Homework4 project in Eclipse

- Right Click over Project Navigator
- Select Import Project
- Browse to the Homework folder
- Click OK

## Complete the Homework

You will need to refer to this handout (Handout4). Make sure you read it thoroughly.

**Before You Begin**

Structure of the Program
Homework4 is an Eclipse Java Project. The Project has

1. A Java Source File called “Homework4.java”
2. A Java Source File called “MyMinecraftPlayerInfo.java”
3. Project references to the Java Runtime Environment 1.6 System Library.
4. Project references to the Minecraft Plugin Library.

Run the Program

1. Click over project name “Homework4”.
2. Toggle the src folder to select Homework4.java
3. Right Click. Select “Run As…”. Select “Java Application”.
   The text “Homework 4” should be displayed in the Console window (bottom)

**Homework Part 1**

Add fields to the class MyMinecraftPlayerInfo

1. Add a field listName to the class MyMinecraftPlayerInfo. The field should be private. The field should have a datatype String.
2. Add a field experience to the class MyMinecraftPlayerInfo. The field should be private. The field should have a datatype float.
3. Add a field foodLevel to the class MyMinecraftPlayerInfo. The field should be private. The field should have the data type int.
4. Add a field onGround to the class MyMinecraftPlayerInfo. The field should be private. The field should have the data type boolean

**Homework Part 2**

Add a constructor to the class MyMinecraftPlayerInfo

1. The constructor will have 4 inputs
   a. initialListName
   b. initialExperience
   c. initialFoodLevel
   d. initialOnGround
2. Add the code in the constructor to make sure that the object state is set correctly to the constructor inputs

**Homework Part 3**

Add methods to the class MyMinecraftPlayerInfo

1. Add a method getListName so that all other parts of the program can access the value of field listName.
2. Add a method getExperience so that all other parts of the program can access the value of the field experience.
3. Add a method getFoodLevel so that all other parts of the program can access the value of the field foodLevel.
4. Add a method isOnGround so that all other parts of the program can access the value of the field onGround.
5. Add a method addFoodLevel.
   The method should take 1 input of type int.
   The method should change the foodLevel by the input if the input value is > 0.
   The method should not change the foodLevel if the input is <= 0.
   The method signature will look like below

```
public void addFoodLevel(int value)
```

**Homework Part 4**

Add the code to in the main method of Homework4.java to

1. Create a variable myPlayerInfo of datatype MyMinecraftPlayerInfo. The variable should have an initial value to be an instance of MyMinecraftPlayerInfo with
    a. listName set to “Player1”
    b. experience set to 10.0
    c. foodLevel to be 4
    d. onGround  to be true
2. Call the method getFoodLevel on myPlayerInfo and print the result to the console using System.out.println
3. Call the method addFoodLevel on myPlayerInfo with 6 as the input
4. Call the method getFoodLevel on myPlayerInfo and print the result to the console using System.out.println

## Submit the Homework

Make sure you are on the Home WiFi.

Follow the instructions in “Submitting Homework” in this Handout.