TINKER ACADEMY

Programming Using Java

Handout 5: Inheritance, Packages and Interfaces Part I

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”.
   You should see a message “Update course in progress”
   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. Wait for a few minutes (to allow for the update to complete)
5. 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 StarterPack5.zip
We will be using StarterPack5.zip for this class.

Click on StarterPack5.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 StarterPack5.zip
StarterPack5 is an Eclipse Java Project. The Project has

1. A Java Source File called “StarterPack5.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 “StarterPack5”.
2. Toggle the src folder to select StarterPack5.java
3. Right Click. Select “Run As…”. Select “Java Application”.
   The text “Starter Pack 5” 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”.

Refresher

Visual Model

The visual model below below shows the blueprint for MyMinecraftHouse and the object myHouse based on the blueprint from Handout4.

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

Refresher Questions

Question	Your Answer
The class is called	
Is this class a new data type	
The datatype is called	
A class defines the _____ and _____ of objects of that type	
myHouse is a	
Of datatype	
A constructor is used to	
How many fields does this class have	
How many methods does this class have	
What is the initial value of name	
What is the initial value of height	

Fundamentals of Java Inheritance

What is **inheritance**?
A class is a blueprint that is used to create **objects** of that type.

The blueprint defines the state and behavior of objects of that type.

Inheritance is a **language mechanism** which allows one class to base its state and behavior off another class.

The class that inherits from another class is called the "child" class or the "sub" class or the "derived" class.

The class that is being inherited from is called the "parent" class or the "super" class or the "base" class.

The inheritance used in Java is called class based inheritance.

The objective of class based inheritance is code reuse.

A Java class can inherit state and behavior from a single class. Such an inheritance is called single inheritance. This is different from C++, which allows state and behavior to be inherited from multiple classes. The inheritance used in C++ is called multiple inheritance.

The code below defines a new class MyMinecraftWoodHouse that inherits from MyMinecraftHouse.

As you will see later, the class MyMinecraftWoodHouse has a blueprint similar to MyMinecraftHouse but will be used to build a house made of wood.

1. Open Eclipse
2. Click project name "StarterPack5" in Package Explorer
3. Right Click, New, Class
4. Type in MyMinecraftWoodHouse for the "Name"
5. Click on "Browse…" for the Superclass
6. Type in MyMinecraftHouse for "Choose a Type".
7. Select the highlighted MyMinecraftHouse class and click OK.
8. Enable the checkbox "Constructors from Superclass"
9. Click on Finish
10. A new Java Source File MyMinecraftWoodHouse.java should be created under "src", "(default package)"

11. Double click MyMinecraftWoodHouse.java to open the file in the Java editor.
12. The class should have a single class MyMinecraftWoodHouse with the following class signature

```
public class MyMinecraftWoodHouse extends MyMinecraftHouse
```

13. Open Problems View. You should see no Errors.

What is the **extends** keyword?

The extends keyword is an optional clause that appears in a class signature and specifies the superclass of this class.

The extends keyword in the class signature for MyMinecraftWoodHouse indicates that the subclass MyMinecraftWoodHouse extends from the superclass MyMinecraftHouse.

The new subclass MyMinecraftWoodHouse is referred to as “inheriting” from the superclass MyMinecraftHouse.

MyMinecraftHouse is the “parent” class or the “super” class or the the “base” class.

MyMinecraftWoodHouse is the “child” class or the “sub” class or the “derived” class.

The extends keyword is optional in the class signature.

Every class except the class Object has a superclass. If the extends keyword is not specified in the class signature, then the superclass of that class is the class Object.

Can a Java Class inherit **any** other class?
A Java class can extend any other Java class as long as

- The class it extends has a constructor that has only private access
- The class it extends does not have the special final modifier

The final modifier in a class signature indicates that the class can never be extended.

Visual Model
The visual model below below shows the blueprint for MyMinecraftWoodHouse.

Class	Method 0	Method 1	Method 2	Method 3
MyMinecraftWoodHouse	Constructor			
Superclass	Field 1	Field 2	Field 3	
MyMinecraftHouse				

Class	Method 0	Method 1	Method 2	Method 3
MyMinecraftHouse	Constructor	build		
	Field 1	Field 2	Field 3	
	name	width	height	

Making Sense of Code Reuse

As mentioned earlier in this Handout, MyMinecraftWoodHouse has a blueprint similar to MyMinecraftHouse but will be used to build a house made of WOOD.

The only part of the blueprint that changes between MyMinecraftHouse and MyMinecraftWoodHouse is that MyMinecraftWoodHouse would need to build a house made of WOOD.

The rest of the blueprint can stay the same.

By the same mechanism, we could define a class called MyMinecraftGlassHouse that has a blueprint similar to MyMinecraftWoodHouse but will be used to build a house made of GLASS.

Again, the rest of the blueprint can stay the same as that for MyMinecraftHouse.

Likewise, we can define classes that make houses made of SANDSTONE or some other material.

Declaring the field with a different name in the subclass
A subclasses can declare a field with a different name from the superclass fields.

For example, the subclass MyMinecraftWoodHouse can have a field “torchOverDoor” of datatype boolean to keep track of whether the TORCH material is over the door.

Declaring new methods in the subclass
A subclasses can declare a method with a different name from the superclass methods.

For example, the subclass MyMinecraftWoodHouse can have a method isTorchOverDoor that returns true if the TORCH material is over the door.

Declaring the field with the same name in the subclass
A subclass can have a field with the same declared as in the superclass. However this field will now hide the field in the superclass. The only way to access the field in the superclass with the same name is by using the super keyword.

Declaring the method with the same name in the subclass
A subclass can declare a method with the same name as in the superclass. This is called method overriding. However, the method signature must be similar to that in the superclass. There are specific rules about overriding methods. For now we will use the simple rule that the method declarations in both classes must have the same signature.

Making Things Abstract

The keyword abstract in a class signature indicates that the class is incomplete or should be considered incomplete.

A class is considered to be incomplete if it has the abstract keyword.

Java does not allow creation of an object from an incomplete class.

What are **class modifiers**?

Class modifiers are keywords that modify a class.

abstract is a class modifier that indicates that the class is incomplete

final is a class modifier that indicates that no subclasses can extend the class

The keyword abstract in a method signature indicates that the method is not defined in this class. Subclasses **should** define the method.

What are **method modifiers**?

Method modifiers are keywords that modify a method.

abstract is a method modifier that indicates that the method is not defined in the class.

final is a method modifier that indicates that the subclass cannot change the method definitio

The code below makes the class MyMinecraftHouse abstract using the abstract class modifier. The code also makes the build method abstract using the abstract method modifier.

This will ensure that the only houses with valid materials get created.

1. Double click MyMinecraftHouse.java to open the file in the Java editor.
2. Add the abstract class modifier to the class signature. The change is shown below

```
public abstract class MyMinecraftHouse
```

3. Add the abstract method modifier to the method signature of the build method. The change is shown below

```
public abstract void build();
```

4. Open Problems View. You should see no Errors.

As indicated earlier the keyword abstract in a method signature indicates that the method is not defined in this class. Subclasses **should** define the method.

MyMinecraftWoodHouse is a subclass of MyMinecraftHouse and now needs to have a method definition for the build method.

1. Double click MyMinecraftWoodHouse.java to open the file in the Java editor.
2. The code below defines the build method build in MyMinecraftWoodHouse. The method has the same signature has that in MyMinecraftHouse but does not have the abstract method modifier

```
public void build() {
    System.out.println("building a house made of " + Material.WOOD);
}
```

3. Save your changes
4. Open Problems View. If you see errors indicating that "Material cannot be resolved to a variable", do the following

a. Click on Material in the Java Editor
b. Right click, Source, Add Import

We will be implementing the rest of the build method for the class MyMinecraftWoodHouse in the next Handout.

Putting it all together

So far, we have done the following

1. Reused the blueprint for the MyMinecraftHouse class for the MyMinecraftWoodHouse class
2. Modified the blueprint so that the MyMinecraftWoodHouse has its own build method that will be used to build a WOOD house
3. We can now use the same pattern to define a class that will be used to build a GLASS house or to define a class that will be used to build a SANDSTONE house.
4. Modified the blueprint for MyMinecraftHouse so that it is marked incomplete. This will ensure that all houses get built using valid material.

Visual Model
The visual model below below shows the blueprint for MyMinecraftWoodHouse.

Class	Method 0	Method 1	Method 2	Method 3
MyMinecraftWoodHouse	Constructor	build		
Superclass	Field 1	Field 2	Field 3	
MyMinecraftHouse				

Class	Method 0	Method 1	Method 2	Method 3
MyMinecraftHouse	Constructor	build		
	Field 1	Field 2	Field 3	
	name	width	height	

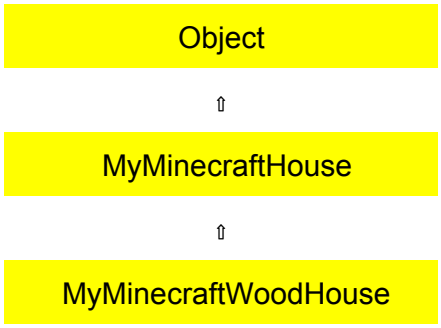
Another Visual Model

The visual model below shows block diagrams of both the classes. The arrow points from the subclass to the superclass. The arrow indicates that the MyMinecraftWoodHouse inherits from MyMinecraftHouse.

Such a diagram is called a UML class diagram.

What is **UML**?
UML stands for Unified Modeling Language.

Its a common language to represent models of object oriented software.



Subclass Privileges

Subclasses have special privileges.

Privilege #1 Constructor Access Privilege

They get to access the superclass constructor using the super keyword.

The code below will change the access modifier of the fields of the class MyMinecraftHouse from private to protected.

1. Double click MyMinecraftWoodHouse.java to open the file in the Java editor.
2. The constructor for MyMinecraftWoodHouse is as below
   In the code below, the constructor for MyMinecraftHouse is invoked using super

```
public void MyMinecraftWoodHouse(String name, int width, int height) {
    super(name, width, height);
}
```

Privilege #2 Field and Method Access Privilege

They get to access a field of the superclass using the super keyword if the field has an access modifier that allows access.

They get to access a method of the superclass using the super keyword if the field has an access modifier that allows access.

Which **access** modifiers allow access to a superclass field or method? The following access modifiers allow access ● public ● protected We have already covered the public access modifier which allows ANY class to access the field or method. In this Handout, we will cover the protected access modifier.

The code below will change the access modifier of the fields of the class MyMinecraftHouse from private to protected.

1. Double click MyMinecraftHouse.java to open the file in the Java editor.
2. Change the access modifiers of the fields by replacing private with protected

   The field name will change from

   ```
   private String name;
   ```

   to

   ```
   protected String name;
   ```

   Make similar changes to the fields width and height.

3. Open Problems View. You should see no Errors.
4. Double click MyMinecraftWoodHouse.java to open the file in the Java editor.
5. Update the build method to that below. As you can see the subclass can access the field name, width and method that are defined in the superclass since these fields have protected access modifiers in the superclass.

```
public void build() {
    System.out.println("building a house made of " + Material.WOOD  + " called " +
name + " of width " + width + " and height " + height);
}
```

6. Save your changes
7. Open Problems View. You should not see any errors.

Privilege #3 Variable assignment

Objects that are created using the new operator from subclasses can be assigned to a variable of the superclass datatype

The code below will change an object from MyMinecraftWoodHouse using the new operator and assign it to a variable of datatype MyMinecraftHouse.

1. Double click StarterPack5.java to open the file in the Java editor.
2. Add the code below at the end of the main method (before the ending curly brace })

```
MyMinecraftHouse myHouse = new MyMinecraftWoodHouse("Casa Rosa", 10, 10);
myHouse.build();
```

3. Right Click over StarterPack5, Run As Java Application
4. The following should be printed to the console

```
building a house made of WOOD called Casa Rosa of width 10 and height 10
```

That was a LOT we covered!

You made it this far! Awesome!

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

Quiz 5: Inheritance, Packages and Interfaces Part I

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 Quiz5.odt under “Courses” “TA-JAV-1” “quiz” “quiz5”

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 5: Inheritance, Packages and Interfaces Part I

Make sure you read this Handout!

Overview

In this Homework you will define a new class MyMinecraftGlassHouse that will reuse the blueprint from MyMinecraftHouse similar to MyMinecraftWoodHouse.

Open the Homework

Follow the instructions in “Updating the Course” in this Handout.
Extract Homework5.zip under “Courses” “TA-JAV-1” “homework” “homework5”

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

Import the Homework5 project in Eclipse

- Right Click over Project Navigator
- Select Import Project
- Toggle on General, Existing Projects into Workspace
- Browse to the Homework5 folder
- Click OK

Complete the Homework

You will need to refer to this handout (Handout5). Make sure you read it thoroughly.

Before You Begin

Structure of the Program
Homework5 is an Eclipse Java Project. The Project has

1. A Java Source File called “Homework5.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 “Homework5”.
2. Toggle the src folder to select Homework5.java
3. Right Click. Select “Run As…”. Select “Java Application”.
   The text “Homework 5” should be displayed in the Console window (bottom)

Homework Part 1
Define a new class MyMinecraftGlassHouse. The class will be similar to MyMinecraftWoodHouse and should extend MyMinecraftHouse.

Homework Part 2
Define the build method for MyMinecraftGlassHouse. The build method will be similar to MyMinecraftWoodHouse but will print GLASS instead of WOOD.

See the section “Privilege #3 Variable assignment” for the method definition for the build method in MyMinecraftWoodHouse

Homework Part 3
Add the code in the main method of Homework5 to

- use the new operator to create an object of datatype MyMinecraftGlassHouse with
  - name “Glass House”
  - width 10
  - height 10
- assign it to a variable “myHouse” of datatype MyMinecraftHouse
- invoke the build method on myHouse

See the section “Privilege #3 Variable assignment” on similar code added to the main method in StarterPack5.

Run the Homework5 main method by selecting Homework5.java, Right Click, Run as Java Application.

The application should print the following

```
Homework 5
building a house made of GLASS called Glass House of width 10 and height 10
```

Submit the Homework

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