

Java Wormhole

God Bennett

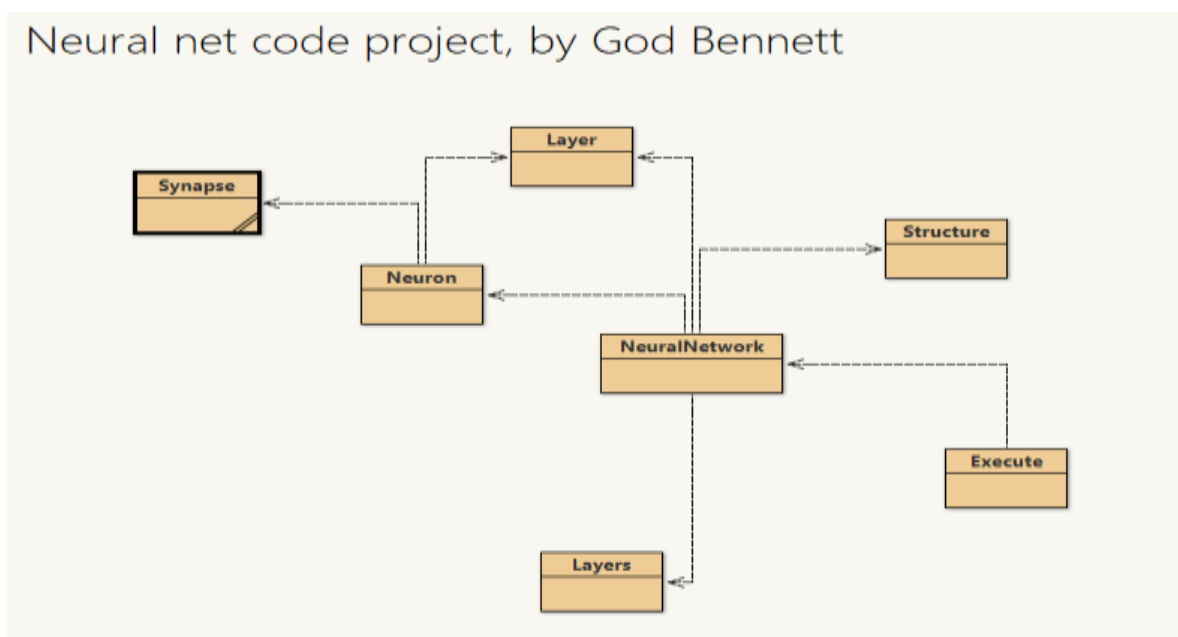
UAD – God Bennett’s “Java Wormhole”

30 minutes to 1 hour: Reasonably rapid movement from 0 java practice to absorption of Java Programming, for the purpose of Universal Ai Diploma

Introduction

As a pedagogical tool, in Java/BlueJ, UAD | Universal Ai Diploma contains a fundamental **artificial neural network programming session**, that grants intuition in candidates regarding the use of complicated machine learning/data science libraries, that normally hide away a majority of the Ai work in the background.

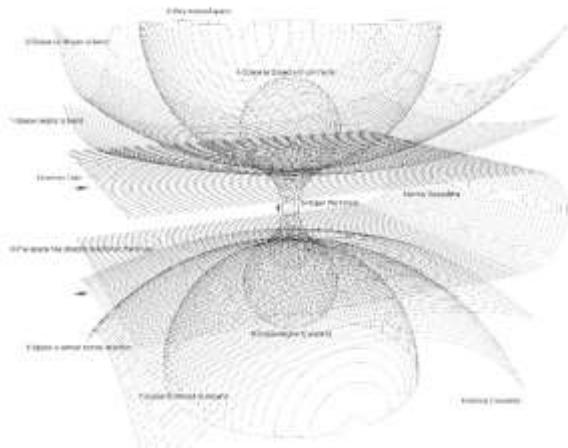
Particularly, BlueJ/Java is an apt way to show **how a neural network’s components connect through the use of** visual maps of how code units relate (where code units are described as partial and main realities on page 4 and beyond):



Why learn fundamental neural networks?

1. Libraries typically hide away lots of work, be it Ai libraries or otherwise, but for eg, [Microsoft's Joseph Albahari notes for example, in his C# Neural Network tutorials](#), understanding fundamental neural networks gives rise to intuition in the usage and debugging of ml libraries such as tensorflow.
2. Beyond debugging, [as underlined by UAD Lecturer God Bennett](#), it is quite empowering to store these ~1000 lines of fundamental neural network code in one's memory, i.e. artificial neural networks are an approximation of our own biological brains!

Java Wormhole – Begin!



Imagine yourself as the creator of a universe. Programming normally consists of

1. **Blueprints/Partial Realities** (i.e your **blueprints/plans** for stuff in your universe)
2. **Reality** (i.e. where you run **instances** of your blueprints/plans)

All programming essentially makes use of Objects/**Blueprints Partial Realities** as well as "Object/**Reality**" i.e somewhere to see those **blueprints** **doing things**, i.e. **the scripts/character descriptions in a TV show** can be likened to these **blueprints/plans**, while **the tv show being broadcast** can be likened to "**reality**" where those **scripts or plans** are show those characters in action or "**instantiated**".

Artificial Neural networks, are essentially loops that expose their structure to supervised pairs of data or examples related to a task/objective, while making use of **Blueprints** and **Reality** (i.e. somewhere to run instances of the objects that comprise the neural network)

Our sample project

Blueprints/Partial Realities: Planet, Tree, Human ← Main **Reality**

Our Sample Project: Java point of view

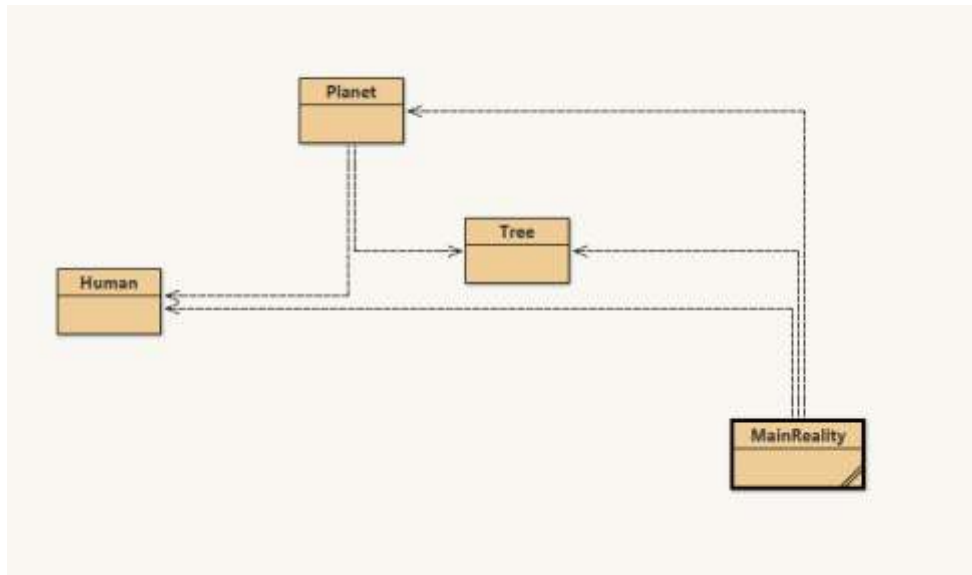
Blueprints/Partial Realities (Classes in Java): Planet, Tree, Human ← Main Reality (Main Class where blueprints are shown in action)

Typically, in programming, for a project, we normally have **partial realities/blueprints** and one **main reality** where all **blueprints** are shown in action through a final “screen”, **the main reality**.

Any coding project we do typically consists of:

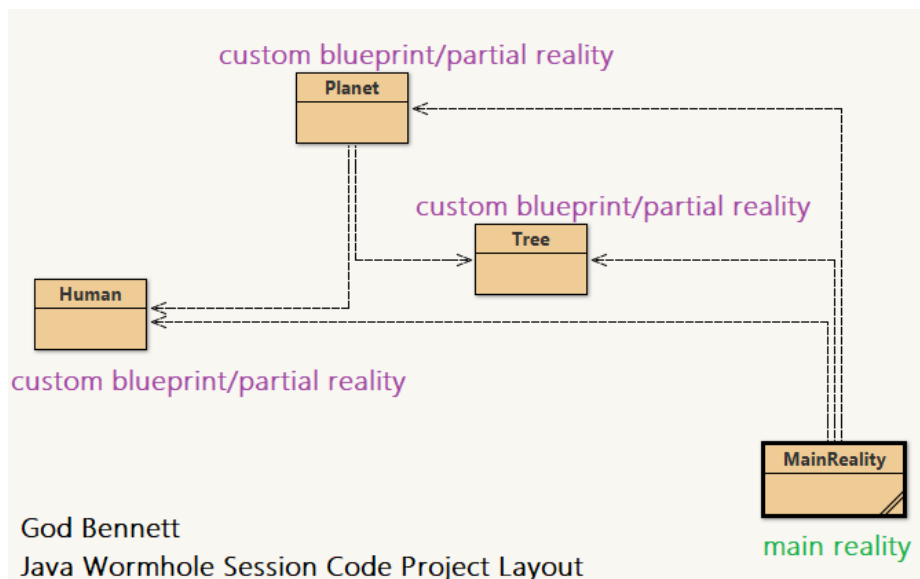
- a. A combination our own **custom-classes/blueprints/”partial realities”** with in-built **classes/blueprints**, specified in the programming language. These can be likened to partial realities, because we call blueprints specified in the language where they are “called to action” in our blueprints.
- b. A **main reality** where everything we build/refer to above are shown in action.

Our Sample Project: Java Code Map



Our Sample Project: Java Code Map (Annotated)

Blueprints/Partial Realities (Classes in Java): Planet, Tree, Human ← Main Reality (Main Class where blueprints are shown in action)



Our Sample Project: Java Code Map (Blueprint/partial reality sample code)

Typically, each custom blueprint will have:

1. Features (characteristics/variables, i.e. human name, id
2. Constructor (For eg: Tells us how to put a human on a planet or in Main Reality, by describing a name)
3. Methods (For eg: Tells us what we can do with a human on a planet or in Main Reality, for eg, getting data – getName() about human is an example of what we can do with a human)



```
//Author: God Bennett
//Universal AI Diploma
//Java_Worldwide - Reasonably rapid movement from 0 java practice to absorption of Java Programming, for the purpose of Universal AI Diploma

//////////OVERVIEW
//1. Describe human blueprint

//////////
//HUMAN BLUEPRINT
//////////

//import Random "Blueprint" into our Human Blueprint
import java.util.Random; //This is an in-built blueprint/class/partial reality (Particular function: Get random number);
//How to use:
//Step 1: Use "import" a built in Blueprint at top of current class/blueprint. eg: "import java.util.Random
//Step 2: Describe Blueprint inside of class/blueprint like below: new Random ( ).nextInt ( )

public class Human
{
    //features
    private int id = new Random ( ).nextInt ( );
    private String name;

    //Constructor
    //Tells us how to put a human on a planet or in Main Reality, by describing a name.
    public Human ( String name )
    {
        this.name = name;
    }

    //methods
    public int getId ( )
    {
        return id;
    }
}
```

Class compiled - no syntax errors.

Our Sample Project: Java Code Map (Main Reality sample code)

Similar to partial realities, or main realities can have features (the partial realities), and methods, including a main function which forms our main screen, or other methods like “System.out.println (“message here”) for revealing data about our partial realities.

```
MacReality - Universal AI Diploma Java Warehouse
Class Edit Tools Options

//Author: Dan Barrett
//Universal AI Diploma
//Java Warehouse - Massively rapid movement from 0 java practice to absorption of Java Programming, for the purpose of Universal AI Diploma

//OVERVIEW
//1. Describe trees, and a list of trees
//2. Describe Humans, and a list of humans
//3. Describe planet based on the above
//4. Observe data about planet/trees and humans using loops

//MAIN REALITY
//Describe trees, and a list of trees
//Describe Humans, and a list of humans
//Describe planet based on the above
//Observe data about planet/trees and humans using loops

import java.util.ArrayList; //This is an in-built blueprint/class/partial reality
//how to use:
//Step 1: Use "import" a built in blueprint at top of current class/blueprint, eg: "import java.util.ArrayList";
//Step 2: Describe blueprint inside of class/blueprint like below: private ArrayList <Tree> trees = new ArrayList<Tree> ( );

public class MainReality
{
    public static void main ( String [ ] arguments )
    {
        //Describe trees, and a list of trees
        //Describe Humans, and a list of humans
        //Describe planet based on the above
        //Observe data about planet/trees and humans using loops

        Tree tree1 = new Tree ( "brown" );
        Tree tree2 = new Tree ( "black" );
        Tree tree3 = new Tree ( "red" );

        //Fire list from trees described
        //Empty container! Contains the use of tree blueprints and standard ArrayList to make an empty container, which we will fill up in our
        ArrayList <Tree> trees = new ArrayList<Tree> ( );
        //Now to fill empty container of trees
        trees.add ( tree1 ); //add function comes standard with ArrayList, although it is an ArrayList of our custom Tree Blueprint
        trees.add ( tree2 ); //add function comes standard with ArrayList, although it is an ArrayList of our custom Tree Blueprint
        trees.add ( tree3 ); //add function comes standard with ArrayList, although it is an ArrayList of our custom Tree Blueprint
    }
}
```

Result after executing our main reality:

