Teacher's Introduction

Programming is a difficult subject to learn and thus is naturally a difficult faculty area to deliver to a high standard. Programming involves maths, a logical mindset and an enthusiasm to problem solve, and if programming is not difficult enough, learners will often start with quite a varied ability of these attributes already. In the delivery of a programming curriculum, it is common within the first few lessons for learners to become fragmented; this is because some learners will demonstrate a natural programming ability and others will not. This poses an obvious problem with delivery and if the fragmentation is not managed correctly, some learners will cease to learn and/or others will not fulfil their full potential; differentiation is essential in programming curriculums. Luckily, programming naturally lends itself to individualisation and therefore it is quite easy to differentiate the learning process to meet distinct learner needs. Hopefully, this resource will help too!

The Programming with Java Resource

This resource is directed at curriculums aiming to deliver computer programming and/or computer science, specifically programming in Java; Java is a good language for beginners to study. The 'Programming with Java' resource spans across KS3 and KS4, with the later chapters being suitable for KS5 lessons. Each chapter contains theory notes, syntax examples, practical activities and a quiz. The theory notes help to place the content being discussed into an appropriate context; the syntax examples help to support this. The practical activities are used to provide a rich programming experience, enabling learners to apply theory to a practical challenge, to confirm their understanding. The quiz recaps on some of the theory learnt from the chapter. The final chapter contains a project for learners to test all skills learnt; this may or may not be suitable, depending on the particular cohort.

How to use the Programming with Java Resource

This resource can be used in many ways, but as with any resource in teaching, it should be tailored to meet specific learning needs. Some suggestions of how this resource may be used are listed below:

- Do not continue to the subsequent chapter until learners confidently meet the learning outcomes of the preceding chapter. This is important as proceeding chapters will recall knowledge learnt in former chapters and will build upon the skills learnt. If learners are not ready to progress, provide them with additional activities to supplement the learning outcomes of the particular chapter.
- This resource is suitable for use in a 'flipped classroom' approach by encouraging learners to read over the chapter theory notes before attending practical classes.
- Differentiation is essential; some learners will complete a chapter prior to others. With a little imagination, however, additional exercises can be prepared that complement the learning outcomes of the chapter. Extension exercises for early finishers will provide teachers with additional time to work with individuals who may have difficulties in comprehending a chapter's content.
- Teachers must be familiar with a chapter's content before the delivery of a lesson; teachers may want to summarise the main outcomes, with code demonstrations, of a chapter collectively before directing learners to work independently.
- The project task in Chapter 10 and other labelled 'hard challenges' are very comprehensive tasks and may not be suitable for all learners. Modify these activities as needed either by simplifying the criteria, providing some of the code pre-written or providing suitable alternatives to the activities.
- Chapters are independently written and thus some students may be comfortable with working through them progressively at their own pace, with little or no interaction with a teacher. This will enable teachers to spend longer with those students who require more support.
- Finally, be realistic in the delivery of programming. Some students will reach higher chapters than others; this is natural. Do not push students onto subsequent chapters simply to keep learning parallel across all students; it is better for a student to understand a little about programming than nothing at all.

The Java Software Environments

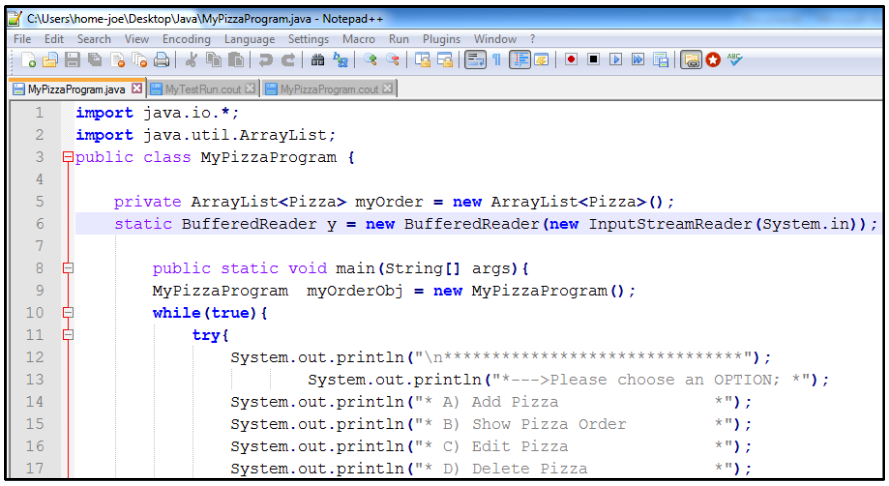
Java can be written in several software environments; perhaps the more popular options are Notepad++, NetBeans and Eclipse. There is no coding environment better than another; each has their own distinct advantages, and choosing an environment is often based on personal preference. For learning purposes Notepad++ can be beneficial as it offers no intelli-sense/debugging (easy identification of errors and/or prompting of available methods/properties when coding), resulting in learners having to examine their own code more closely, as well as to make better use of the Java documentation when authoring code. That being said, some may argue that an environment such as NetBeans, which includes an active intelli-sense/debugging, may develop a learner's ability more quickly, because errors can be easily identified and code can be written more quickly. As a tutor, it would be beneficial to examine all three environments, before making an informed decision as to which is the best environment for a particular cohort. The good news is that all of the environments highlighted are free! Before installing any of the environments, ensure that the Java platform is installed on the host machine.

Notepad++ and the Java Plugin

Notepad++ is a simple text editor which is great for authoring many languages, including Java, HTML and PHP. There is a Java plugin freely available on the web which can be used to speed up the development of Java projects; the plugin offers a facility to run and compile Java files from Notepad++ itself. Other than highlighting the keywords used in Java, Notepad++ offers no other support in writing Java code, which can be beneficial when learning Java. Notepad++ is free and can be downloaded from the following link: http://notepad-plus-plus.org/

The Java plugin is also free and can be downloaded from this following link http://sourceforge.net/projects/npcompilejava/

Once the plugin is downloaded, unzip the folder and copy and paste the content into the 'plugin' directory, found in the Notepad++ install folder. Sometimes installing the Java plugin can be tricky as the appropriate environment variables may need to be set. However, a quick Google search will help resolve this issue.

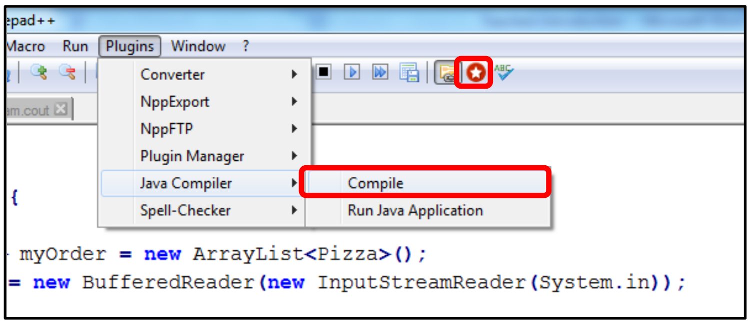

C:\Users\home-joe\Desktop\Java\MyPizzaProgram.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
MyPizzaProgram.java
MyTestRun.cout
MyPizzaProgram.cout

```
import java.io.*;
import java.util.ArrayList;
public class MyPizzaProgram {

    private ArrayList<Pizza> myOrder = new ArrayList<Pizza>();
    static BufferedReader y = new BufferedReader(new InputStreamReader(System.in));

        public static void main(String[] args){
        MyPizzaProgram  myOrderObj = new MyPizzaProgram();
        while(true){
            try{
                System.out.println("\n*******************************");
                        System.out.println("*--->Please choose an OPTION; *");
                System.out.println("* A) Add Pizza                *");
                System.out.println("* B) Show Pizza Order         *");
                System.out.println("* C) Edit Pizza               *");
                System.out.println("* D) Delete Pizza             *");
```

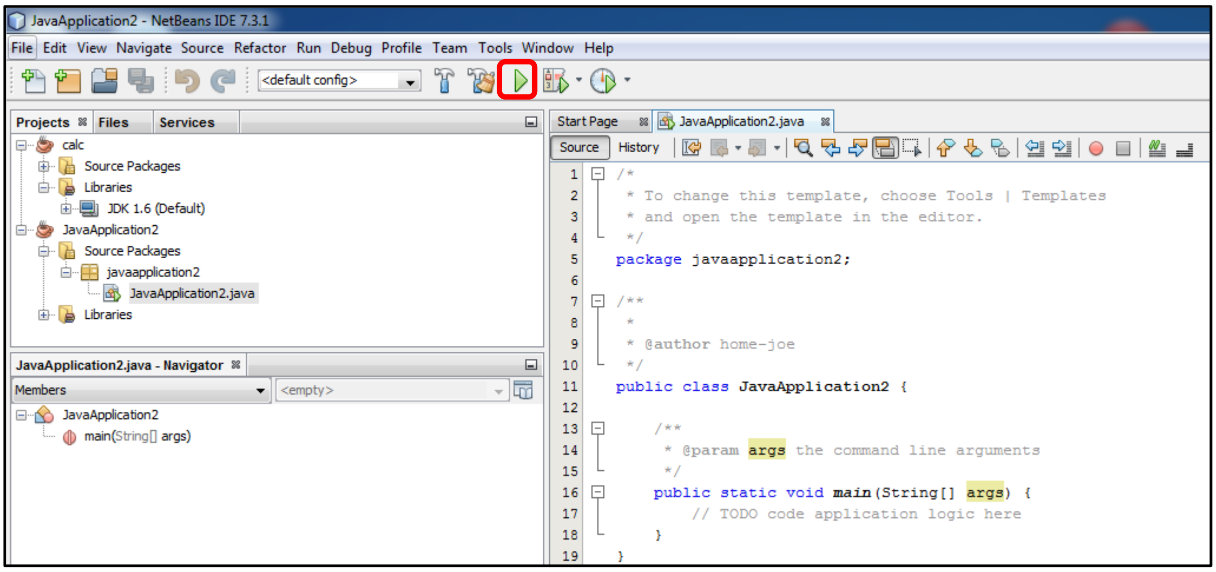
Once the Java plugin is set up and points to the javac compiler, it is ready to use. To compile a program, first save the file as a '.java' file with the same name as the Java class. Also, ensure that the class contains the 'main' method. Next, click on Plugins → Java Compiler → Compile. The first time a program is compiled, Notepad++ will prompt the user to create a '.count' file, click 'Yes.' Finally, once the program is compiled, it can be run by clicking the red and white star button, found on the toolbar. Don't forget to save and recompile the program whenever any new changes are made to the program code.



```
{
 myOrder = new ArrayList<Pizza>();
= new BufferedReader(new InputStreamReader(System.in));
```

NetBeans

NetBeans is an integrated development environment (IDE) for developing Java and thus comes loaded with many tools for developing comprehensive software projects. One distinct advantage of using NetBeans over Notepad++ is that the program will automatically highlight errors as the code is written, whereas Notepad++ will rely on the errors to be highlighted by the compiler. Other useful tools include the project structure (found in the left window pane) and the intelli-sense which will prompt the user with possible suggestions when authoring code. Another useful feature associated with the IDE, as opposed to Notepad++, is that programs can be quickly compiled and executed in a single step, by simply clicking on the green 'run' button. NetBeans is a free environment which can be downloaded from the following link: https://netbeans.org/



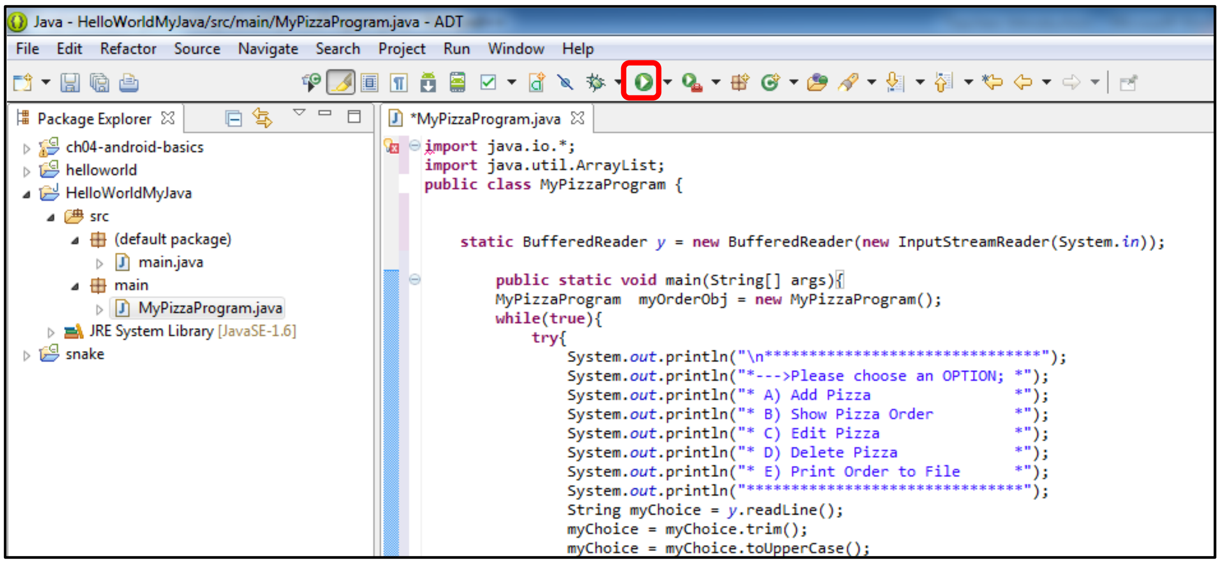
```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package javaapplication2;

/**
 *
 * @author home-joe
 */
public class JavaApplication2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

Eclipse

Eclipse, like NetBeans, is another integrated development environment (IDE) used for developing Java and thus Eclipse too comes bundled with many tools for developing comprehensive software projects. In fact, Eclipse has many similar features to NetBeans, including the project structure, the highlighting of coding errors, the built-in intelli-sense and the green 'run' button (which will compile and execute the program in a single step). As stated earlier, it is largely personal preference when determining which environment to use. Eclipse is free and can be downloaded from the following link: https://www.eclipse.org/downloads/


Java - HelloWorldMyJava/src/main/MyPizzaProgram.java - ADT
File Edit Refactor Source Navigate Search Project Run Window Help
Package Explorer
ch04-android-basics
helloworld
HelloWorldMyJava
src
(default package)
main.java
main
MyPizzaProgram.java
JRE System Library [JavaSE-1.6]
snake
*MyPizzaProgram.java

```
import java.io.*;
import java.util.ArrayList;
public class MyPizzaProgram {


    static BufferedReader y = new BufferedReader(new InputStreamReader(System.in));

        public static void main(String[] args){
        MyPizzaProgram  myOrderObj = new MyPizzaProgram();
        while(true){
            try{
                System.out.println("\n*******************************");
                System.out.println("*--->Please choose an OPTION; *");
                System.out.println("* A) Add Pizza                *");
                System.out.println("* B) Show Pizza Order         *");
                System.out.println("* C) Edit Pizza               *");
                System.out.println("* D) Delete Pizza             *");
                System.out.println("* E) Print Order to File      *");
                System.out.println("******************************");
                String myChoice = y.readLine();
                myChoice = myChoice.trim();
                myChoice = myChoice.toUpperCase();
```