CSci 1933 Lab 1 **(September 15, 2015)**

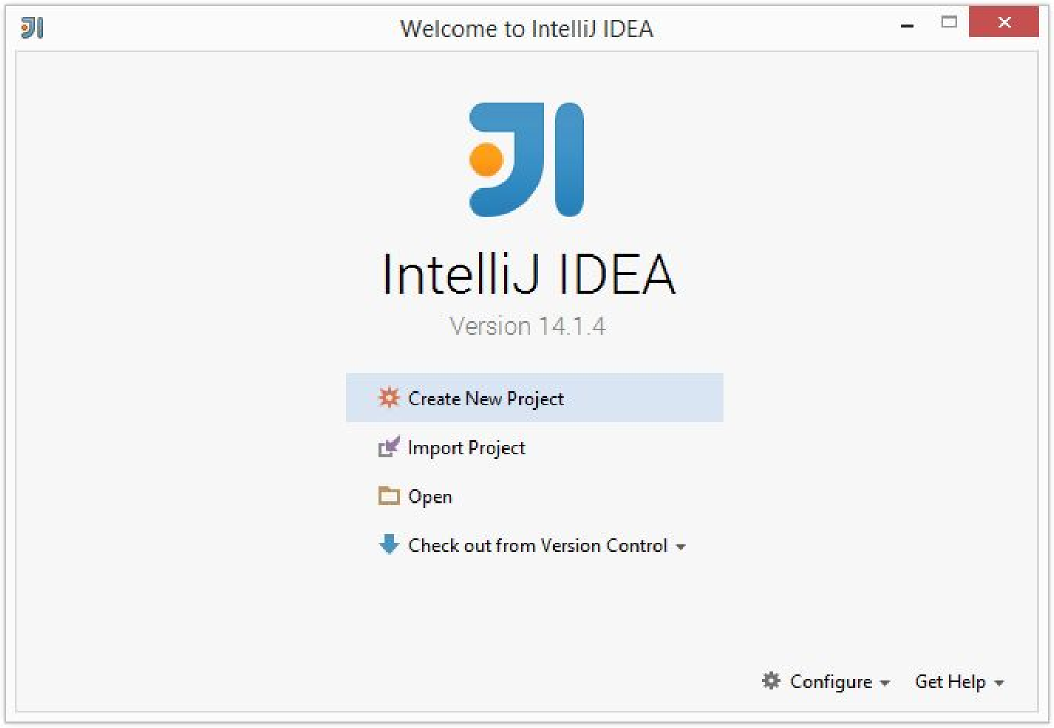
Today's objectives:

- How to start and run JAVA projects
- Getting familiar with IntelliJ (an IDE)
- Start working with primitive and class types

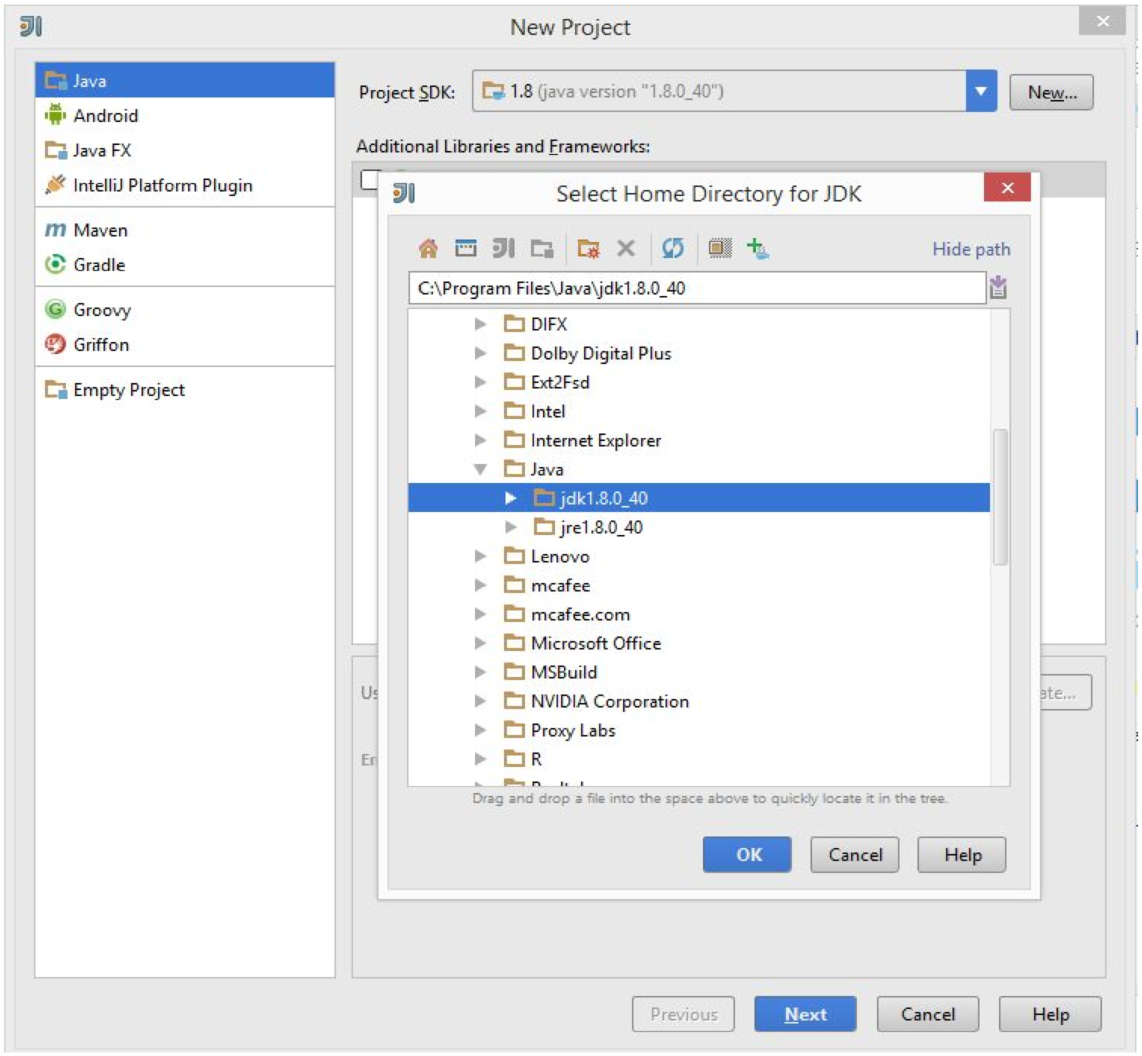
1. Creating a project in IntelliJ

Any new development in IntelliJ IDEA starts with creating a project. To create a project:

- Open IntelliJ through command line (Ctrl + Alt + T)
- Once the IntelliJ window opens, select "Create new project" option.

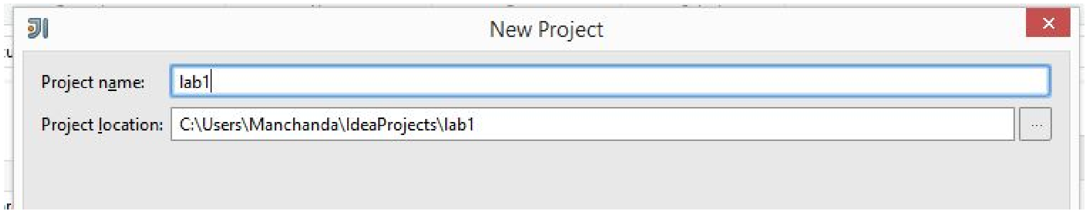


- In the new window that appears, in the left-hand pane, select Java. (We want a Java-enabled project to be created). In the case <None> is shown the Project SDK field on right panel, the JDK is not yet specified. You should specify it now by clicking New and selecting JDK. In the Select Home Directory for JDK dialog that opens, select the directory in which you have the desired JDK installed (ask TA for help), and click OK. Then click Next (See the screenshot below)



- The options on the next page will help us to create a Java class with a main() method. However, we will do the very same from scratch So, don't select any of the options. Click Next.

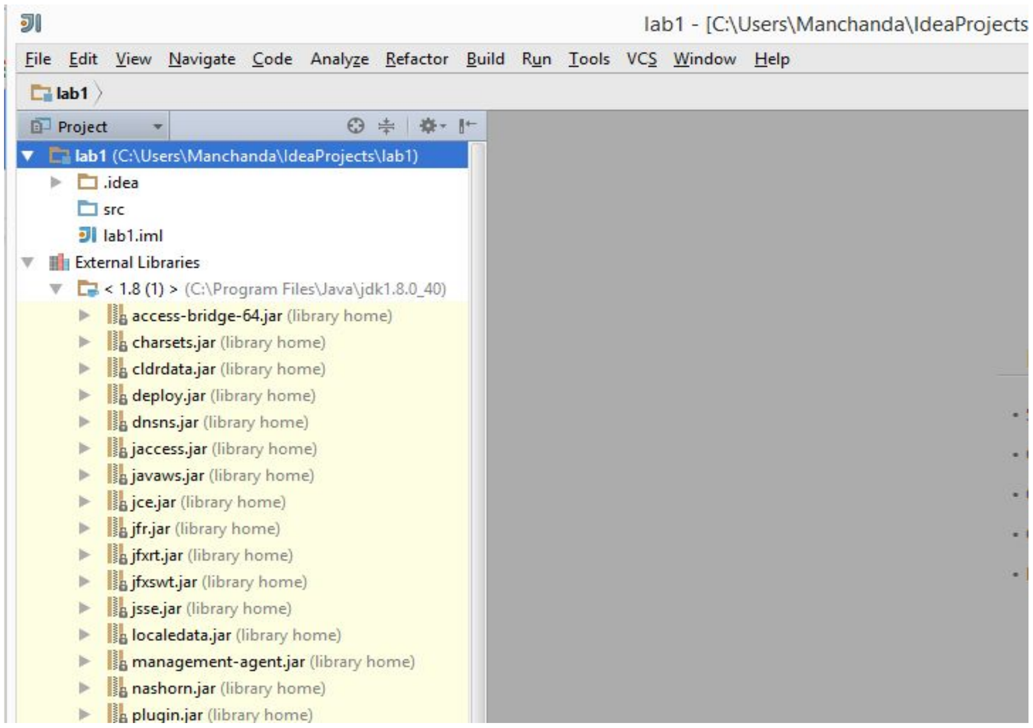
- On the next page, specify the project name (e.g. lab1). Click Finish.


New Project

Project name: lab1

Project location: C:\Users\Manchanda\IdeaProjects\lab1

- IntelliJ will take some time to complete project creation process. Once the process is complete, the structure of your new project is shown in the Project tool window. There are two top-level nodes:
    - lab1- This node represents your Java module. The .idea folder and the file lab1.iml are used to store configuration data for your project and module respectively. The folder src is for your source code.
    - External Libraries- This is a category that represents all the "external" resources necessary for your development work. Currently in this category are the .jar files that make up your JDK.


lab1 - [C:\Users\Manchanda\IdeaProjects
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab1
Project

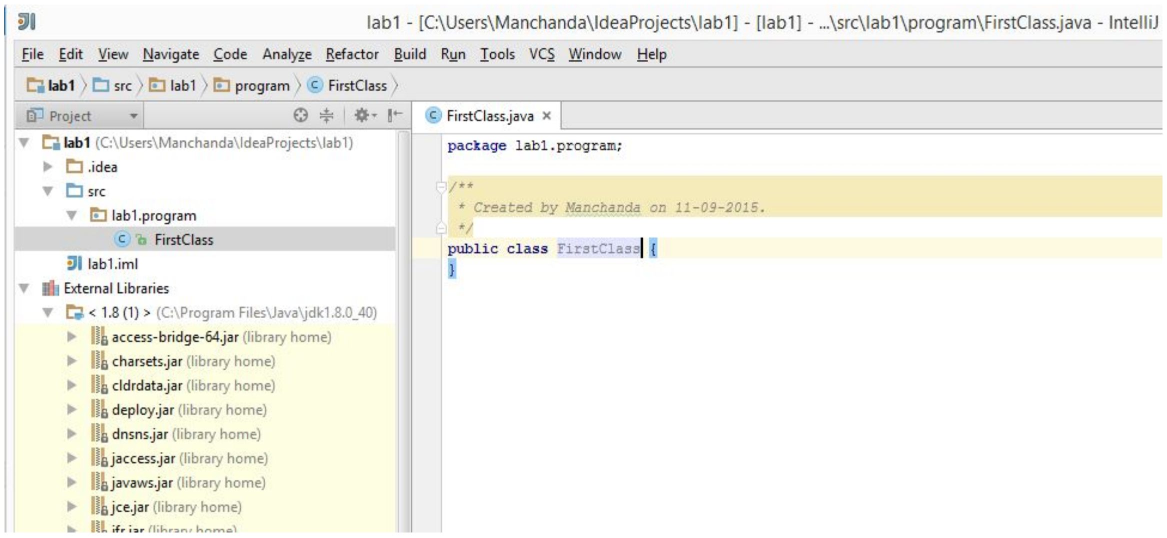
```
lab1 (C:\Users\Manchanda\IdeaProjects\lab1)
    .idea
    src
    lab1.iml
External Libraries
    < 1.8 (1) > (C:\Program Files\Java\jdk1.8.0_40)
        access-bridge-64.jar (library home)
        charsets.jar (library home)
        cldrdata.jar (library home)
        deploy.jar (library home)
        dnsns.jar (library home)
        jaccess.jar (library home)
        javaws.jar (library home)
        jce.jar (library home)
        jfr.jar (library home)
        jfxrt.jar (library home)
        jfxswt.jar (library home)
        jsse.jar (library home)
        localedata.jar (library home)
        management-agent.jar (library home)
        nashorn.jar (library home)
        plugin.jar (library home)
```

- Congratulations, you have successfully created a JAVA project.

2. Programming in JAVA for the first time.

Java is an object oriented language. Consequently, the building blocks of a JAVA program are classes. A package is a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being similar to different folders on your computer. You might keep HTML pages in one folder, images in another, and scripts or applications in yet another. Because software written in the Java programming language can be composed of hundreds or thousands of individual classes, it makes sense to keep things organized by placing related classes into packages. In this section, we will learn how to create packages and classes in JAVA. To begin with, we'll create a program that will print something to the screen.

- Now we are going to create a package and a class. Let the package and the class names be edu.csci1933f2015.lab1 and FirstClass respectively. In the Project tool window, select the src folder and select File | New, or New from the context menu for the folder src. In the New menu, select Java Class. In the Create New Class dialog that opens, type edu.csci1933f2015.lab1.FirstClass in the Name field. The Class option selected in the Kind list is OK for creating a class. Click OK. The package edu.csci1933f2015.lab1 and the class FirstClass are shown in the Project tool window. At the same time, the file FirstClass.java (corresponding to the class) opens in the editor.


lab1 - [C:\Users\Manchanda\IdeaProjects\lab1] - [lab1] - ...\src\lab1\program\FirstClass.java - IntelliJ
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
lab1 〉 src 〉 lab1 〉 program 〉 FirstClass 〉
Project
FirstClass.java ×
lab1 (C:\Users\Manchanda\IdeaProjects\lab1)
.idea
src
lab1.program
FirstClass
lab1.iml
External Libraries
< 1.8 (1) > (C:\Program Files\Java\jdk1.8.0_40)
access-bridge-64.jar (library home)
charsets.jar (library home)
cldrdata.jar (library home)
deploy.jar (library home)
dnsns.jar (library home)
jaccess.jar (library home)
javaws.jar (library home)
jce.jar (library home)

```java
package lab1.program;

/**
 * Created by Manchanda on 11-09-2015.
 */
public class FirstClass {
}
```

- Now, whenever a JAVA program is executed, before anything else, the control straightaway goes to a method (function) called “main”. In this step, we’ll create “main” method in our FirstClass class. The code to create main method in FirstClass is shown below. Please note that the keywords public, static are special JAVA keywords. We’ll learn about them in future lectures. For now, just copy and paste.

```java
public class FirstClass {
  public static void main(String[] args) {
  }
}
```

- System.out.println is a Java statement that prints the argument passed, into the System.out which is generally stdout. We will use this statement to print something to the screen. Since, this will be the first task we need to do, we’ll put this as our first statement in main method we created in previous step. See the code snippet below:

```java
public static void main(String[] args) {
  System.out.println("Go UMN! Beat Kent State!");
}
```

  To execute the code, right-click somewhere in the editing area and select Run 'FirstClass.main()'. Wow, something is printed on output screen.
  Note the double quotes around “Go UMN! Beat Kent State!” in System.out.println statement. Think about why it is required. Try to execute this code without these double quotes. Also note the semicolon at the end of print statement. In JAVA, every statement is ended with a semicolon.

- Now, we’ll work will some primitive data types in JAVA. First, we’ll declare and initialize int and boolean variables and then print them using System.out.println(). To declare and initialize a variable in JAVA, the syntax is:

```
[modifiers] type identifier [=value];
```

where,

- [modifiers] -> optional – These are special Java keywords like public, static etc. Leave them for now. We'll discuss them later lectures.
- type -> required – This specifies the type of information or data held by the variable. ex: int, boolean etc can be either primitive or can be object reference type.
- identifier -> required – This is the name that you assign to the variable.
- [=value] -> optional – This is an initial value that you may want to assign to the variable. It should match with the type of the data that we have specified. Note that although it is optional, every variable should be used before it can be used anywhere else in the program.

The variable declaration/initialization should end with semicolon because it is a statement.

First, let us declare and initialize an int variable. An example declaration and initialization is:

```
int a = 1;
```

Here, 'a' is the identifier with value 1. Declare and initialize an int variable similar to above and place that statement in our main method after our last System.out.println() statement. Now, print the value of variable 'a' using the System.out.println() statement.

Should the required statement be System.out.println("a") or System.out.println(a)? Try both, execute the program and guess the difference.

In JAVA, anything written in double quotes(“”) is considered a string. So System.out.println(“a”) prints the string “a” while System.out.println(a) prints the value of variable a.

Now, you should be able to answer why we got error in last section when we tried to execute System.out.println(Go UMN! Beat Kent State!); (Note, without the double quotes), because our system tried to find out value of variable Go UMN! Beat Kent State!. However there was no such variable present, in fact, it is not even a valid variable name (Note the red line under it in editor), so it threw an error. A valid JAVA variable name must follow certain rules. Variable names are case-sensitive. A variable's name can be any legal identifier — an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "$", or the underscore character "_". The convention, however, is to always begin your variable names with a letter, not "$" or "_". Additionally, the dollar sign character, by convention, is never used at all. A similar convention exists for the underscore character; while it's technically legal to begin your variable's name with "_", this practice is discouraged. White space is not permitted. Subsequent characters may be letters, digits, dollar signs, or underscore characters.

Next, let us declare and initialize a boolean variable. Note that a boolean variable can be initialized as either true or false. An example declaration and initialization is:

```
boolean b = true;
```

As in last case, also print this boolean variable using System.out.println() statement.

- Now, we’ll work will some objects in JAVA. Compared to primitive data-types, apart from declaration and initialization, we also need to instantiate the objects. To declare, initiate and initialize a variable in JAVA, the syntax is:

```
[modifiers] type identifier = new type( [arguments]);
```

The new keyword is a Java operator that creates the object.

First, we'll declare and initialize String and Integer variables and then print them using System.out.println().

First, let us declare, instantiate and initialize a String variable. An example is:

```
String s = new String("Go UMN!");
```

Again note the double quotes in the case of String. Print this String variable using System.out.println() statement.

Next, let us declare, instantiate and initialize an Integer variable. Note that while int is a primitive datatype, Integer is a class. So, their syntax differs.

```
Integer i = new Integer(6);
```

As in last case, also print this Integer variable using System.out.println() statement.

2. Define your own class and create its objects

So far, we have worked with primitive datatypes and objects of predefined classes. Now, we'll create our own class and work with it.

- Create a new class as we did before that is, in the Project tool window, select the src folder and select File | New, or New from the context menu for the folder src. In the New menu, select Java Class. In the Create New Class dialog that opens, type edu.csci1933f2015.lab1.FootballTeam in the Name field. Note that the package name is same as before and the class name is FootballTeam. The class FootBallTeam will be shown in the Project tool window. At the same time, the file FootballTeam.java (corresponding to the class) opens in the editor.

- FootballTeam class will have two variables. A string called name with value "Gophers" and an int called numPlayers initiated with any value. (You choose yourselves). See the snippet below to understand. Like before, just copy and paste keywords public, final etc.

```
public class FootballTeam {
  public final String name = "Gophers";
  public final int numPlayers = 11;
}
```

- Now, our class is ready. We will now create an object of FootballTeam. To do this, move back to main method in FirstClass class file. We'll create a object called "team" of class type FootballTeam using the syntax we learnt in last section.

  ```
  FootballTeam team = new FootballTeam();
  ```

  Next, we'll print the variables in this FootballTeam object using System.out.println() method. In JAVA, to access the data stored in object, we use following syntax:

  ```
  object.data;
  ```

  For eg., team.name will return a String called "Gophers".

  Our next task is to execute following line of code:

  ```
  System.out.println("The " + team.name + " has " + team.numPlayers + "
  players");
  ```

  Execute the above code and see the output. Now, you can guess the use of '+' operator between strings.

  Finally, our FirstClass will look something like this:

  ```
  package edu.csci1933f2015.lab1;

  /**
  * Created by Manchanda on 11-09-2015.
  */
  ```

```
public class FirstClass {
  public static void main(String[] args) {
      System.out.println("Go UMN! Beat Kent State!");
      int a = 1;
      System.out.println(a);
      boolean b = true;
      System.out.println(b);
      String s = new String("Go UMN!");
      System.out.println(s);
      Integer i = new Integer(6);
      System.out.println(i);

      FootballTeam team = new FootballTeam();
      System.out.println("The " + team.name + " has " + team.numPlayers + "
players");
  }
}
```

<u>This ends our Lab 1</u>

IntelliJ Reference:
https://www.jetbrains.com/idea/help/creating-and-running-your-first-java-application.html

JAVA Reference:
https://docs.oracle.com/javase/tutorial/
https://tecnoesis.wordpress.com/2009/11/07/declaring-initializing-and-using-variables/