In this worksheet you will go through creating classes, basic input and output, use of primitive types, the string/array/vector class, and control flow.

1 Introductory

1.1 First Steps in the Java Development Environment

Step 1: Open up the IDE and lookup "Create a Class". Name this class Demo1 for your first Java program.

A. How do you create a class?

Step 2: Now create a main method to provide an entry point into your program. The syntax to do so is:

```
public class Demo1 {

   /* Send out a welcome message */
   public static void main(String[] args) {

      //comments in Java are the same as C++
      System.out.println("Welcome to CS202's Java Workshop \textbackslash n" + "Enjoy!");
   }
}
```

Whoa! Public static void main(String[] args) let's break this down a little.

A. Public means the this method is accessable from outside the class.

B. static means the method is owned by the class and not an instance (object). This allows main to be called without ever instantiating an object of the encapsulating class.

C. main returns void now instead of int. The System object has an "exit status" field that is returned by the Java Virtual Machine when the program terminates.

D. The array of Strings as an argument is optional but is recomended. This array will contain the arguments provided on the command line (space delineated). Unlike C and C++, the first item in this array is NOT the name of the program.

Step 3: The System object provides an interface to access the output and error streams, the current time, and other useful tools like array copy or exit(). Take a look at Oracle's documentation of this object and see what is available.

Step 4: 'out' is a field of the System object of type PrintStream. System.out.println(to_print) does exactly what you would expect. Changing println to just print prints without a following new line. PrintStream also provides methods to flush the buffer and check for errors.

Step 5: Compile and run the program in the console window.

A. Shift F10 or Run — Run Main

B. Or, use the green "run" arrow from the toolbar by debug

1.2 Creating Classes and Methods

Step 1: With the exception of a couple differences, loops, conditionals, and primitive data types are the same as C++ with the addition of some syntactical options if you prefer. Some key differences to keep in mind:

A. boolean is spelled out now

B. if statements must evaluate to Truth or False, unlike C++ where zero/NULL would be false and non-zero would be true

C. null is now lowercase

Step 2: Write the following method in the Demo1 class:
The method will take three integer arguments. The method will display all the numbers, each on their own line, that are divisible by the third argument and inbetween the first two arguments. Remember that you need the scope specifier at the start of every method. Write the function signiture below:

Step 3: Test this class by calling it from main. Make sure it works as expected! Next we will add I/O utility so the user can interact with this method.

1.3 Creating a Utility Base Class for Generalized I/O and Changing the Program Entry Point

Step 4: Next we will create a utility base class that any class that needs user input can inherit from.

A. Create a new class called Utility.

B. There are many methods of working with input in Java. We will use a Scanner object. Scanners can be attached to an external data file but Java provides a FileReader class that is worth looking at.

C. At the top of this new file for the Utility class import the Scanner utility:

```
import java.io.*
import java.util.Scanner
```

D. Give the Utility class a field for a Scanner referance and write a constructor for the class that initializes it:

```
public class Utility {
   protected Scanner input;
   public Utility() {
      input = new Scanner(System.in);
}
```

Step 5: Let's move main() outside of the Demo1 class to clean things up a little bit. Do this by creating another class called Main and move the main function to this class. You will have to edit the entry point of your program to your new main() to run your program. (Click Run on the taskbar dropdowns and click Edit Configurations. There you will see where you can change the main class)

2 Intermediate

2.1 Using the I/O Utility class, the String Class, Class Fields, and Arrays

Step 1: First take a look at the Scanner and String classes. Go to the docs to see what Scanner methods are available. Some scanner methods that will be useful include:

A. input.nextInt(); //returns the next whole number

B. input.nextFloat(); //returns the next float

C. input.next(); //returns the nest single word as a String

D. input.nextLine(); //returns the next phrase ending in a newline as a String

Step 2: Like in C++, be aware of newlines in the input stream. Use input.nextLine(); to clear a line from the buffer.

Step 3: While reading in some data from the user let's look at some of the functionality of the String class. The String class has many useful methods. Go to the docs and search String. Navigate to the methods overview section and use it to fill in the return and argument types for the following methods (some may have no arguments):

A. _____equals(_____);

B. _____equalsIgnoreCase(_____);

C. _____compareTo(_____);

D. _____compareToIgnoreCase(_____);

E. _____ = String + String;
Why might you want to use the concat() method instead of the overloaded + operator?
(Hint: think efficiency)

_____

F. _____length(_____);

G. _____contains(_____);

H. _____toCharArray(_____);

I. _____toLowerCase(_____);

J. _____clone(_____);

Step 4: Create a new class called Person. Give Person private String fields for first/last name, float fields for hourly_wage/hours_worked, and int fields for age/dollars/quarters/dimes/nickles/pennies. Write a constructor for Person to set these fields to some defaults. (null for the Strings, 0 for ints and floats)

```
public class Person extends Utility {
   private String first;
   private String last;
   private float hourly_wage;
   private float hours_worked;
   private int age;
   private int quarters;
   ...

   public Person() {
      first = null;
      last = null;
      ...
   }
}
```

Step 5: Give the person class a public build() method. Have this method prompt the user to set the values for all the fields. Remember, the scanner methods return the values.

```
this.first = this.input.next();
```

The 'this' keyword isn't required in this context but it is idiomatic Java to use it whenever accessing fields of the current class.

Step 6: Test the build method by creating a person from main() and calling the build function. (from the main class)

```
public class Main {
   public static void main(String[] args) {
```

```
   }
```

Step 7: Now add the following methods to the Person class:

A. display() — displays a Person's details

B. work(float hours) — adds an ammount to a Person's hours_worked field

C. pay() — uses the hours_worked and hourly_wage fields and adds to the dollars and coins fields.

Step 8: Add an array of Strings for performance reports. Set it to null in the constructor. Show how to add this field to the class:

And show how to create an array of 5 strings called reports:

And add a method that adds a performance report for the person by asking the user to input a report. Make sure to take care of the special cases! Here are some tips:

A. The array object has a length field (array.length). This is the CAPACITY of the array, not the number of items in the array. You have to keep track of that yourself with another variable OR iterate through it searching for null referances.

B. You can copy the content of an array using a for loop OR using the System method:
System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length);
See the docs for more details.

Step 9: Last, create a method in Person to help you sort people by their last name. The function should take another person as an argument and return an int. Use the String method compareToIgnoreCase(src) to determine which person comes first.
REMEMBER: Java overloaded the + operator for the String class but not the $==$, $>, <, \leq$, or $\geq$ operators. So do not try to check string equality with $==$. Doing so will compare the referances to the actual Strings themselves.