First java program input/output debugging and datatypes

LegioN

April 29, 2023

Contents

1 structure of a java file

- Name: every file that ends with an extensions .java is a class itself; eg: main.java
    1. since it's name is main.java(or some name before _.java) it should contain a class with that name (as in before the extension) in the file itself
    2. all the code that we'll be writing will be in this class main/name
- the Class in the name: remember if a variable name starts with a capital letter then it is a class name (by convention)
    1. If the class with the name of the file is in the file then that will be a class of the public type , in this case main is the public class type , public class means that this class will be accessible by all other files , classes, etc.

2. Like cpp , java also has a main function (by convention) from where the program starts , whenever we run the program the compiler will look for main , if it is not present then the compiler will throw error.
3. You can run with the java compiler "javac" name.java and then java name (where name is the class as well as the file name) , also while running the executable(class file) with java you can use the class name only.

1.1 explanation of the main.java file

```
public class main { // inside this block of {} is the int main of java as found in cpp
    public static void main(String[] args) {
        System.out.println("hello world!!");
    }
}
```

Here class is name group of properties and functions , in the above eg's class we have this function

1.1.1 *all about the function*

1. All the functions that are in the classes are knows as methods.
2. Also the function name has to be main only like here " public static void main(String[] args) " it is reserved to be here if it is not main it will not run since main function is the entry point of the function. c) Function is a block/collection of code which can be used again and again.

1.1.2 *then rest of the code explanation*

1. Here public means the same as in the class name part " public static void main(String[] args) " since main is necessary to run the program so it makes sense to make it available to be executed from anywhere i.e public class type , if we make it private then it will not be available like the public class and hence it will not be valid.
2. Here static means: this main function/method is a part of the Main class also main is required to be run without creating an object of this Main class ,so we use the static type . Since nothing is running before the main() function then there is no use of making an object there.

3. Static variables and functions are the variables and functions that don't depend on the object.

4. void is the return type of the function, since we don't want any value here so we have used void(like in cpp).

1.1.3 *how to run*

1. (String[] args) are the arguments , collection of strings.

2. here args is the arguments given in the terminal with java command (like java Main 50 100 . . .) after compiling with the javac(like javac Main(class/file name)) and the no inside the [] is the index of the array to which the values given with the java command will be printed , the values given with the java command is stored in the String[] array.

3. if you use javac -d directory name: this -d flag is used to give a directory to store the .class file.

4. package is the folder where the java file will lie, eg package com.kunal , this com.kunal is a subfolder in the com folder (you can make more of those subfolders) , '.' period means subfolder.

5. then comes = System.out.println("hello world!!"); = this prints a string , if we use ctrl/cmd+click on this we get.

```
/**
 * Prints a String and then terminate the line.  This method behaves as
 * though it invokes {@link #print(String)} and then
 * {@link #println()}.
 *
 * @param x  The {@code String} to be printed.
 */
public void println(String x) {
    if (getClass() == PrintStream.class) {
        writeln(String.valueOf(x));
    } else {
        synchronized (this) {
            print(x);
            newLine();
        }
    }
}
```

6. to print stuff we have the System class , it is in file called System.java(since System starts with a capital letter hence we know it is a class) in the java.lang package. All the stuff available in the lang package can be accessed in the files created by the people who made java

7. println in the above src says that you will give a string and it'll print that string

8. out is a variable contained in the System class, it is basically a type of PrintStream(will be expained later), and PrintStream has something called println. out is like a reference variable for PrintStream and by default value of the standard output stream i.e the out is the commandline i.e laptop (idk what that means) or the place where the output will be shown, out's original value is null (like this here) = public static final PrintStream out = null; which means that the output will be in the commandline but if we declare out = some file or anything then it will take the output of that println into that file or something

9. ## so it means System has a variable called out which is of type PrintStream and this out has a method/function called println, since out is of type PrintStream so println is also in PrintStream : in the standard output stream print something

10. **NOTE** some of the reference which can be added to the code itself instead of here are done in the file mentioned here

11. for input in Main.java the code: ~ Scanner input = new Scanner(System.in); ~ Scanner is a public class (it allows us to take input in simple sense) which is a simple text scanner which can parse primitive types and strings using regular expressions, then the input is a normal declared variable and by using Scanner , this input will read everything we want ,then we add a new object i.e 'new' which is a keyword then Scanner(), in the brackets here we need to pass from where we will pass the input (can be a file or keyboard), we pass the System.in which is used to pass standard input or output (.in_ part refers to the keyboard input) also the default value of System.in is null we can also define it to be something else . object is a type of scanner and it has a value of System.in whenever input asks for something you will take the value from the keyboard this is the meaning ; again scanner is basically a class that specifies an input stream and using the variable (object) of the class we can take input. *also look for the main documentation in the IntellijIdea by ctrl+click*

- better explanation of the **System** class: The System class contains several useful class fields and methods. It cannot be instantiated. Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.
- and of the **.in** part: The "standard" input stream. This stream is already open and ready to supply input data. Typically this stream corresponds to keyboard input or another input source specified by the host environment or user. In case this stream is wrapped in a java.io.InputStreamReader, Console.charset() should be used for the charset, or consider using Console.reader().

12. regarding the new keyword , it is used to
13. every class in java extends the object class , eg: Scanner ; important concept
14. about the different next(,Int,ln) variants

```
System.out.println(input.nextInt()); // = nextInt prints the integer outpu
System.out.println(input.next()); // = next prints the first string(/word)
System.out.println(input.nextln()); // = nextln prints the whole string of
```

input is a variable pointing to the object of scannner class so it'll contain all the functions provided by the scanner class ; whatever input will be given it will be shown in output

1.2 All about the datatypes

1. Primitive data types are the ones which cannot be further broken into other datatypes . eg: "Kunal" this word can be further divided into individual letters i.e the string can be divided into char , but the char cannot be further divided into other data types, this last datatype which cannot be divided into further data type is known as primitive
2. **NOTE** To know why we are adding f to float and L to long at their respective end of the data type declaration value , we need to know the size of the bitwise operators (also the range of them will be in the bitwise operators), for now the size are : int and float has 4 bytes, double and long has 8 bytes

(a) All the decimal values are of the type double by default so if there is a need to store them in float then we need to add an 'f' (remember lower case f only) after the end of the value *as to why use double instead of float: float gives floating point errors sometimes by rounding off numbers, so to get more accurate value we use double* also we can store larger decimal values using the double type

(b) And the by default declaration type of integer values is "int" so we use 'L' (remember upper case L only) at the end of the long type, *why use long instead of int: because it can store more/long integer values*

3. There contains a class for every data type known as wrapper classes for giving additional funtionality to the primitive datatypes, and they are written as

```
Integer rollno = 8;
String name = "tera baap hu";
```

(a) as you can see it contains the capital first letter which by convention means classes so all the other datatypes are written like the above example (more on that later on OOPs)

(b) how to comment: you can add single line comments using "/" and multi line comments using the following

```
/* this is a multi line
comment
*/
```

1.3 explanation for the inputs.java file (some are in the file itself)

1. you can use debugging to see how each line is executed one by one, also add breakpoints to make sure which lines you want to debug

2. about literals and identifiers:

```
int a = 10;
```

- here 10 is a literal, so in primitive datatypes literals are the syntactical representation of datatypes, i.e it can be used to represent specific values of a datatype and

- and a, the reference variable is known as the identifier, so it's basically the name of the identifier, variable, class, packages, or other stuff like interfaces in java, etc

3. int values can be given in a long form by adding some underscores (eg $100_{10009707}$) which gets ignored while running

4. nextFloat() takes float values, just like that you can end the after next___ underline part with the datatype which you want to get some input about, see completion for some help

5. + is used to join the sentences or strings together in "" as well as with the value of the variable (as shown in the sum2nos.java file)

1.4 explanation for the sum2nos.java file (some are in the file itself)

```
public static void main(String[] args){
    Scanner num = new Scanner(System.in);
    float input = num.nextFloat();
    System.out.println(input);
}
```

1.4.1 TypeCasting and TypeConversion

When we give int values in float datatype and after getting the output we see the result as 105.0 (for c = 5).Because the int input is being converted to float, this is auto type conversion.

1. So when one type of data is assigned to another type of variable, then auto type conversion will take place if the following conditions are met

   (a) the two types should be compatible, for ex = int and float, etc.

   (b) the destination type should be greater than the source type be it in the declaration value or the input value, for ex = here int is smaller than float so float converts to int but vice versa is false. Asking for int but giving float won't work since int<float

   (c) java does auto type conversion when it stores an integer constant into a variable of types like byte, short, long, and char sometimes , takes the ascii value of that

- so how to convert int to float, this is known as typecasting or casting in compatible types. It is used as the auto type conversion is helpful but it might not fulfill all the needs, for example - if you want to assign an integer variable to a byte variable or a float value to an integer variable(this will not happen auto, since they are greater than the other). This is sometimes called narrowing conversion since it tells the float value to narrow down the float value to integer