**Java-Java: 02 - Variables**

**Lesson:**

Variables are the way that data is stored in Java. There are two types of Variables in Java: Primitive and Reference. In Java, some of the Primitive types are: int, double, float, boolean, and char (We will go into what each of these types are in a moment). Reference types are all other types of variables, including all objects. A good example is the String type. An easy way to tell the difference between a Primitive and Reference type is that Primitive types are all lowercase while Reference types use camel case (the first letter of each word is in uppercase). When variables are created in Java, their tag is linked to a memory address on the Stack (for our purposes just think of this as a block with many separated memory addresses). The difference between Primitive and Reference types comes from the way they are stored in memory.

**Primitive Types**

The data from Primitive types is stored directly on the Stack. This means that at the memory location linked to a Primitive variables tag, is the data the variable is storing.

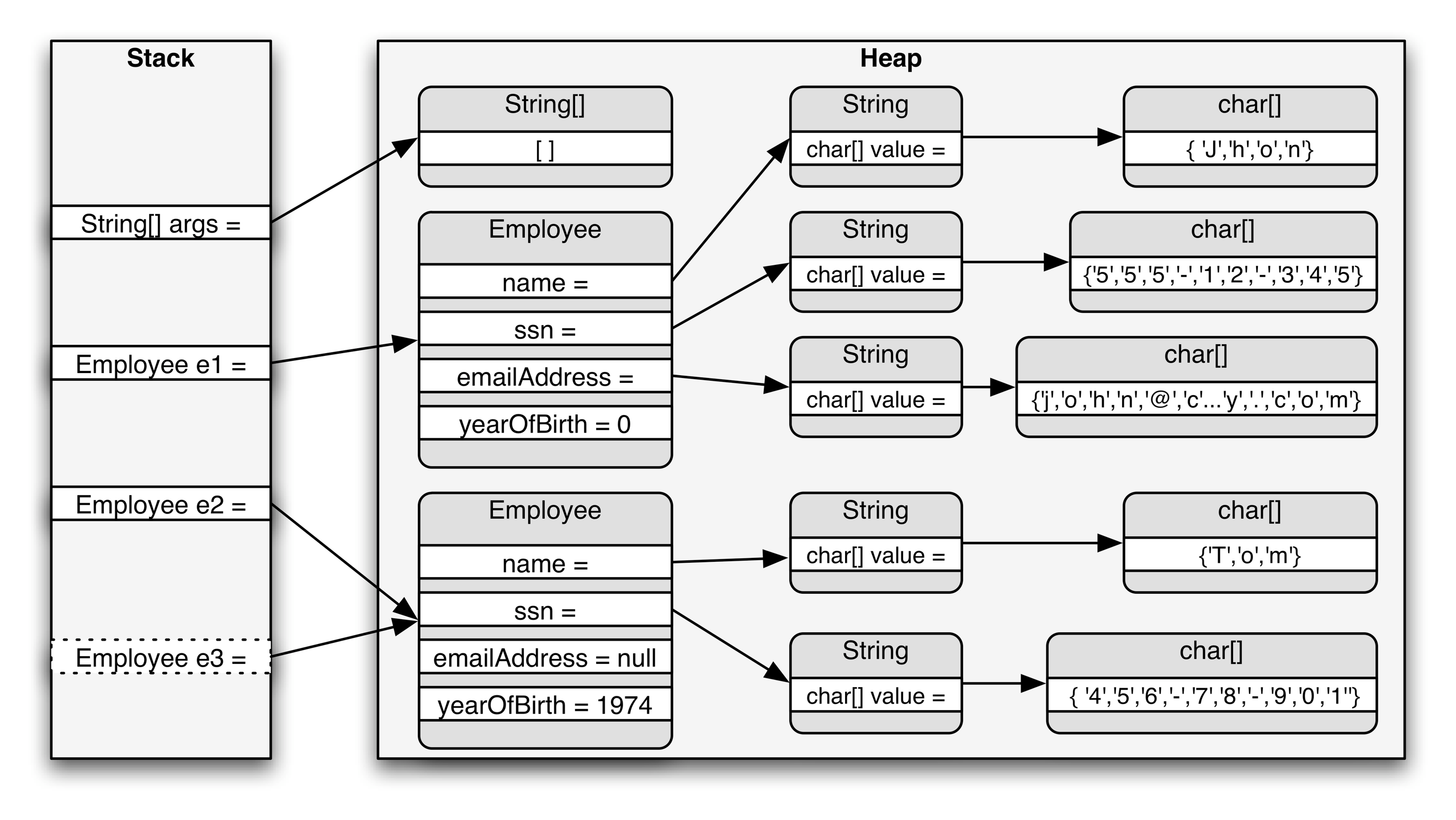
Now let’s go into what each Primitive type of data can hold. The table below contains the type and what it stores.

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Example** |
| int | int, more commonly known as integer, holds numbers that are mathematical integers. These are whole numbers. | -3, -2, -1, 0, 1, 2, 3 |
| float | A float holds any number, including ones with decimals. | 1, 1.0, 3.14, -36.24 |
| double | A double holds numbers, including ones with decimals. | 1, 1.0, 3.14, -36.24 |
| boolean | A boolean is either True or False. | True, False |
| char | A char, also known as a character, is any single letter, symbol, or number. All chars are inside a set of single quotes (‘ ‘) | ‘A’, ‘3’, ‘!’, ‘a’ |

As you may have noticed, float and double seem to hold the same type of data. The difference between them is the amount of bytes of memory allocated to them. A float only has 4 bytes while a double has 8. This means that a double can store a larger number, in terms of total number of digits. A int is also only 4 bytes but due to only storing integer values, it can represent a much larger range of numbers.

**Reference Types**

A Reference type stores it’s data on the Heap (essentially the same as the Stack but data does not have to be stored in consecutive blocks of memory). The memory address linked to a Reference types tag on the Stack holds a Reference to the first memory address allocated to the variable on the Heap. While this may sound confusing, it is not too important to know the difference behind the scenes between how Reference and Primitive types are stored (If you are interested, below is a diagram that shows the difference within memory. Additionally, this was a simplified explanation, there is more detail to how memory works in Java that we will go into later).



An example of a reference type is all Objects. We will go into Objects later so don’t worry too much about it now. A reference type that will be commonly used is String. Notice the capital ‘S’, this signifies that the type is an Object (you may have also noticed that all the primitive types were lowercase). We will go more into the proper capitalization in Java later, along with other Java style guidelines.

**Creating Variables**

All variables are created in a process of two steps: Declare and Initialize. The declare step is the same for primitive and reference types, while the initialize step differs between them.

**Declaration**

Declaring variables is the step where a location in memory is linked to a tag. Declaring variables is incredibly easy. All you need to do is follow the formula:

*Type* *Tag*;

When creating a variable you replace *Type* with the datatype you want the variable to be, and you replace *Tag* with the name you want to associate the variable with. Notice the semicolon (;) at the end of the statement. Semicolons end most statements in Java, as well as in many other languages.

**Initialization**

Initialization is the step where a value is assigned to a variable. In other words, this is when something is put in the memory address linked with the tag. This again is very simple, just follow the formula:

*Tag* = *value*;

The *Tag* is the same as the one from the declaration step. You then set it equal (=) to the *value* you want to put in the variable. This is where the similarities between primitive and reference type variables in initialization ends. For now, we will only focus on primitive type variables and String type variables, since String is a special type of reference variable. We will take a look at reference types again in the Objects assignment.

**Combining Steps**

These two steps can be combined into one line using the following formula:

*Type* *Tag* = *value*;

This is simply the two formulas from above being combined at the *Tag*. There are times when you want to separate the declaration and initialization steps, but for now you can do it either way.

**Examples:**

**Primitives**

package primitives;

public class Primitives {

    public static void main(String[] args)

    {

        //Declaration

        //Type Tag;

        int numberInt;

        float numberFloat;

        double numberDouble;

        char character;

        boolean bool;

        //Initialization

        //Tag = value;

        numberInt = 20;

        numberFloat = 1.25f; //The f differentiates floats from doubles

        numberDouble = 36.24;

        character = 'A'; //Notice the single quotes '' around A

        bool = true; //Notice that true is in blue, this means its a reserved word

        //Print the Variables

        System.out.println(numberInt);

        System.out.println(numberFloat);

        System.out.println(numberDouble);

        System.out.println(character);

        System.out.println(bool);

        //Reassign the Variables

        numberInt = 254;

        numberFloat = 740.0f;

        numberDouble = 6.0;

        character = '$';

        bool = false;

        //Print the Variables

        System.out.println(numberInt);

        System.out.println(numberFloat);

        System.out.println(numberDouble);

        System.out.println(character);

        System.out.println(bool);

        //Notice that the values printed are different now

        //More Variables

        int moreInts = 2;

        double moreDoubles = 44.81;

        //Reassign Variable to another Variable

        numberInt = moreInts;

        numberDouble = moreDoubles;

        //You can also sometimes assign different types to one another

        numberDouble = numberInt;

        System.out.println(numberDouble); //Notice that it added a decimal

    }

}

**Notes:** Some types are able to be assigned to each other. For now we can look at int, double, and float. Of these three, double is the most accepting. It can accept any number within its range (google the range of primitives in Java if you want to know). Floats can be set to an int. An int can’t be set to a double or float without truncating it, also known as removing the decimal part.

**Strings**

package strings;

public class Strings {

    public static void main(String[] args) throws Exception {

        //Declare a String

        String str;

        String name;

        String helloWorld;

        //Initialize a String

        str = ""; //This is the empty String

        name = "Josh";

        helloWorld = "Hello World!"; //Notice that all Strings are surrounded by double quotes ""

        //Print Strings

        System.out.println(str);

        System.out.println(name);

        System.out.println(helloWorld);

        //Reassign Strings

        str = "abcdEFG123!@#";

        name = "Kyle";

        helloWorld = "Goodbye World?";

        //Print Strings

        System.out.println(str);

        System.out.println(name);

        System.out.println(helloWorld);

        //Notice the output changed

        //Set Strings to Each Other

        str = name;

        helloWorld = str;

        //Print Strings

        System.out.println(str);

        System.out.println(name);

        System.out.println(helloWorld);

    }

}

**Notes:** Strings are a reference type but they act in a lot of ways like primitives. This is because they have special code behind the scenes that allows them to act in special ways. We will look into that when we get to Objects.

**Exercises:**

1. Create a program with two variables of type int. Give them any name (tag) you like and set them to any value you like. Then print them out to the console.
2. Create a program with a double and an int. Give them any name you like and set them to any value you like. Then print them to the console. Next change the value of the int and then set the double to the int. Do this by using the tag of the int as the value you are assigning to the double. Then print them out.
3. Create a program with a char. Give it any name you want and set it to any character you want. Then print it out.

**Next Assignment Link:** https://classroom.github.com/a/wacETzVW