Java
John Marchand – jhmarchand@gmail.com
Team 4381 – Twisted Devils
https://github.com/jhmarchand/java4381

Week 5

StringCalculator

Lets look at the stringCalculator

```
	private static int calculateFromString(String equation) {

		int answer = 0;
		char [] equationCharArray = equation.toCharArray();
		String numberString = "";
		int a=0;
		int b=0;
		char opChar = ' ';

		for( int i = 0; i < equationCharArray.length; i++ ) {
			if ( equationCharArray[i] == '+' || equationCharArray[i] == '-'  ||
equationCharArray[i] == '*'  || equationCharArray[i] == '/' ) {
				a = Integer.parseInt(numberString);
				numberString = "";
				opChar = equationCharArray[i];
			}
			else {
				numberString = numberString + equationCharArray[i];
			}
		}

		b = Integer.parseInt(numberString);

		switch ( opChar )
		{
		case '+':
			answer = a + b;
			break;
		case '-':
			answer = a - b;
			break;
		case '*':
			answer = a * b;
			break;
		case '/':
```

```
                answer = a / b;
                break;

        }

        return answer;
    }
```

The for loop walks the chars in our input string saving each one, building up a number. When it hits a operator (+, -, *, or /) it converts that string to a number, and remembers the operator. Then it builds up a second string until the end of the array.

Lets look at a new statement – Switch Case. This is a lot like a big if statement. It allow you to do lots of different things based on a variable matching a specific value. Here we do plus, minus, multiply or divide based on what the operator char was.

Switch statements are not used much, but can be handy, and it good for you to be familiar with them.

Debug

As our programs get larger, sometimes it can be hard to tell why they aren't doing what we want. It's also a good way for you to see how program flow really works. Lets run the debugger on our StringCalculator.

Exercise

Can you finish the GetWordingOfInt function below? So if I pass in 5 it should return "five". Make it work for any number 1 to 9. Use a case statement. Try out the debugger. What happens if you forget the break statement?

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

        int myInt = 5;

        String intAsString = GetWordingOfInt(myInt);

        System.out.println(intAsString);

    }

    static String GetWordingOfInt(int digit)
    {
        String word = "";

        return word;
```

```
	}
```

Classes

So we have ints and Strings as variable types, but did you know we can create our own types called classes? Lets consider a person, they have a name and an age. We can define our own class. Create New class in the menu, name it SimplePerson (don't check the static main box).

```
public class SimplePerson {

	String firstName;
	int age;
}
```

Lets create a main to use this class

```
public class SimplePersonMain {

	public static void main(String[] args) {

		SimplePerson me = new SimplePerson();
		me.firstName = "John";
		me.age = 47;

		SimplePerson luke = new SimplePerson();
		luke.age = 13;
		luke.firstName = "Luke";

		printPerson(me);
		printPerson(luke);

		// have a birthday!
		me.age++;

		printPerson(me);
		printPerson(luke);

	}

	static void printPerson(SimplePerson person){
		System.out.println(person.firstName + " is " + person.age);
	}

}
```

A lot going on here. First, since this isn't a basic type, we have to ask for memory for the computer to store our class variable. That's what new does. So when assigning our classes we say `SimplePerson me = new`

`SimplePerson();`, which in English says I want a variable of type SimplePerson called me and I need a spot in memory to store it. Then we can use its properties to set it up to have my name and age. We can have 2 different people, one for me, one for luke. We can print our the information, even age me a year if its my birthday, but notice this doesn't affect luke. Here's the output from running this.

```
John is 47
Luke is 13
John is 48
Luke is 13
```

Constructors and Class Functions

Not only can we add members (or fields) to a class, but we can add functions. Consider this more complete person class.

```
public class Person {

	String firstName;
	int age;

	Person(String aFirstName, int anAge) {
		firstName = aFirstName;
		age = anAge;
	}

	void haveABirthday() {
		age++;
	}
}
```

In addition to our fields we have 2 functions. haveABirthday() is a function that takes no parameters and returns nothing, but it does something. It ages our person by 1 year.

The Person function is special. A function with the same name as the class is called the constructor. It is a function that is automatically called when you ask for memory. In fact, for our simplePerson class, the compiler generated a default one behind the scenes for us. We have created a Person constructor here that takes 2 parameters that allows us to initialize our Person when we create it. Consider the following code.

```
public class PersonMain {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Person me = new Person("John", 47);
		Person luke = new Person("Luke",13);


		printPerson(me);
		printPerson(luke);

		me.haveABirthday();

		printPerson(me);
		printPerson(luke);

	}

	static void printPerson(Person person){
		System.out.println(person.firstName + " is " + person.age);
	}

}
```

This has the same output as before, but with a few differences, we initialize our class as we create it. Also, we call our haveABirthday function which increments the age.

Exercises

Create a variable for yourself and print out your information.

Can you add a lastName field member to the Person class and use it in PersonMain?

Can you create a class function called GetFullName() that returns a string and use it in PersonMain?

Can you create an array of Persons called family and fill it in for your family?

Can you create a function that returns a Person called GetOldest(Person[] people)?

Can you create your own class called Pet? What members could it have? What functions would it have?