Chap05Yourname.java

Conditional and logic

```
import java.util.Scanner;

public class Chap05 {

    private static Scanner in;

    public static void main(String[] args)  {
        String fruit1, fruit2;
        int m, n;
        double x;
        in = new Scanner(System.in);

    //type here next

}
```

5.1 Relational operators

```
System.out.println("5.1 Relational operators");
m = 5;
n = 5;
System.out.println(m == n); // is equal to
System.out.println(5 != 6); // is not equal to
System.out.println(5 > 3); // greater than
System.out.println(5 <= 5); // greater than or equal to
// There are only two boolean values: true and false
//true and false are boolean values, so they can be displayed directly
```

5.1 Relational operators

```
fruit1 = "Apple";
fruit2 = "Orange";
System.out.println(fruit1.equals(fruit2));
//Most relational operators don't work with strings. We use the
//equals method with String.
```

5.2 Logical operators

```
System.out.println("5.2 Logical operators");
System.out.println(5 > 0 && 5 <= 10); // and
System.out.println(!(5 > 10)); // not
System.out.println(! true);
//5 > 10 should be put into parentheses
System.out.println(true || 5 > 10); //or
//true || anything is always true, so there are dead codes
//after || dead code caused by short circuit
//Likewise, false && anything is always false.
System.out.println(false && 5 < 10);
newLine();
```

5.2 De Morgan’s Laws

```
System.out.println("5.2 De Morgan's Laws");
System.out.println(! true || ! true);
System.out.println(! (true && true));
System.out.println(! false || ! false);
System.out.println(! (false && false));
System.out.println(! true || ! false);
System.out.println(! (true && false));
newLine();
```

5.2 De Morgan’s Laws

```
System.out.println(! true && ! true);
System.out.println(! (true || true));
System.out.println(! false && ! false);
System.out.println(! (false || false));
System.out.println(! true && ! false);
System.out.println(! (true || false));
newLine();
```

5.3 & 5.4 Conditional statements, chaining and nesting

```
System.out.println("5.3 & 5.4 Conditional statements, chaining, and nesting");
checkPosNeg();
checkOddEven();
newLine();
```

5.3 & 5.4 Conditional statements, chaining and nesting

```
public static void checkPosNeg() {
        System.out.print("Type a floating-point number x = ");
        double x = in.nextDouble();
        if (x == 0) {
                System.out.println("x is zero.");
        } else  if (x < 0) {
                System.out.println("x is negative.");
        } else {
                System.out.println("x is positive.");
        }
}
```

5.3 & 5.4 Conditional statements, chaining and nesting

```
public static void checkPosNeg() {
        System.out.print("Type a floating-point number x = ");
        double x = in.nextDouble();
        if (x == 0) {
                System.out.println("x is zero.");
        } else {
                if (x < 0) {
                        System.out.println("x is negative.");
                } else {
                        System.out.println("x is positive.");
                }
        }
} //Chaining and nesting
```

5.3 & 5.4 Conditional statements, chaining and nesting

```
public static void checkOddEven() {
        System.out.print("Type an integer m = ");
        int m = in.nextInt();

        _______________

______________________________________________

______________

_____________________________________________

________
}
```

5.3 & 5.4 Conditional statements, chaining and nesting

```
public static void checkOddEven() {
        System.out.print("Type an integer m = ");
        int m = in.nextInt();
        if (m % 2 == 0) {
                System.out.println("m is even.");
        } else {
                System.out.println("m is odd.");
        }
}
```

5.5 & 5.6 Flag variables and the return statement

```
System.out.println("5.5 & 5.6 Flag variables and the return statement");
printLog();
printLog();
newLine();
```

5.5 & 5.6 Flag variables and the return statement

```
public static void printLog() {
	System.out.print("Type a POSITIVE floating-point number x = ");
	double x = in.nextDouble();
	boolean nonPositiveFlag  =  (x <= 0);
	if (nonPositiveFlag) {
		System.out.println("ERROR: " + x + " is not POSITIVE!!!");
		return;
		//Return statement terminates a method before you reach the end of it.
	}
	System.out.println("The natural logarithm of x is y = " + Math.log(x));
}
```

5.7 Validating input

```
System.out.println("5.7 Validating input");
        printLogPlus();
        printLogPlus();
        printLogPlus();
        newLine();
```

5.7 Validating input

```
public static void printLogPlus() {
        System.out.print("Type a POSITIVE floating-point number x = ");
        in.nextLine();
        //This line is critical.
        if (!in.hasNextDouble()) {
                System.out.println("ERROR: " + in.next() + " is not a NUMBER");
                return;
        }
        double x = in.nextDouble();
```

5.7 Validating input

```
        if (x <= 0) {
                System.out.println("ERROR: " + x + " is not POSITIVE!!!");
                return;
                //Return statement terminates a method before you reach the end of it.
        }
        System.out.println("The natural logarithm of x is y = " + Math.log(x));
}
```

5.8 Recursion

```
System.out.println("5.8 Recursion");
System.out.print("Let us count down from n = ");
n = in.nextInt();
countDown(n);
newLine();
```

5.8 Recursion

```
public static void countDown(int n) {
        if (n == 0) {
                System.out.println("GO!");
        } else {
                System.out.println(n);
                countDown(n - 1);
        }
}
```

5.10 Binary numbers

```
System.out.println("5.10 Binary numbers");
System.out.print("The binary representation of n = ");
n = in.nextInt();
displayBinary(n);
newLine();
```

5.10 Binary numbers

```
public static void displayBinary(int n) {
        __________
_____________________________________
___________________________________________
_________
}
```

5.10 Binary numbers

```
public static void displayBinary(int n) {
        if (n > 0) {
                displayBinary(n / 2);
                System.out.println(n % 2);
        }
}
```