A Complete Java Revision Book By Shantanu Suryawanshi

Shantanu.tk

```
//
Import
package
s
import java.util.Scanner;
import java.util.Random;
// import java.util.*; will import all of java util packages. Import must be done before public class...



// public: this is a "modifier" placed before the word class that makes the class visible (accessible) from outside the class.
//  - In a private class, methods and variables can only be accessed by methods within the same private class
// The class 'revision' must be contained in the file revision.java (the public class and file name must have the same name).
public class revision
{


    // Method signature:
    // static: Forces an instance of the main() method to exist so that it can be called from outside the class.
    // void: this tells us that the main method (function) does not return any values. Return type can be int, String, arrays[], etc.
    // main: this is the method that the JVM looks for and provides the entry point of your program.
    // String: is a sequence of text characters. It is the argument for the main() method. Input parameters.
    public static void main(String args[])
```

```
{


        /* Examples of other method signatures:
                public static void time() // returns nothing, accepts no parameters
                public static void square(int x) // returns nothing, accepts an int number
                public static void name(String fname, String lname) // returns nothing, accepts two strings
                public static String name(String county) // returns a String, accepts a String
                public static char[] result(String s[], int p) // returns char array, accepts a String array and an int number
                public static double whoKnows(char c, int x, double b, String s, String p, int y, float f)
                        // returns a double number type, accepts a char, int, double, string, string, int and a float
        */




        // System.out: is an object used for printing to screen.
        // println: is a method belonging to the System.out class/object that can print a string to the console...
        // Text inside the double quotes is an argument passed to the println() method.
        System.out.println("Hello CS141 classmate!");
        // All statements in Java end with a semi-colon
```

```
        // ----- Declaring variables: <type> <identifier> = <value>;

        int age = 18; // an integer value

        // declaring multiple variables of type integer.
        int day = 0, month = 0, year = 0;

        // the final keyword tells java that this variable cannot be changed later in the program - it is a constant.
        final int myNumber = 7;

        // other variable types to store numbers: byte, short, long, float, double. These can store diffent length of numbers.
        float sampleFloat = 0.0f; // can store decimal numbers with 7 digit precision
        double sampleDouble = 0.0; // can store decimal numbers with 15 digit precision

        char you = 'u'; // stores a single character
        // we must use single quotes around a character value
```

```
boolean validYear = false, validName = false; // stores true or false. Default value if not specified is false.


// A String is a sequence of characters (digits, letters etc). It is anything and everything grouped together between 2 double quotes.
// Notice that String starts with a capital 'S'
String name = "Daniel"; // we must double quotes for strings values
// Another method for declaring a string: String name = new String("Daniel");


String firstName = "", lastName = ""; // initializing variables for later use


/* ---------- Relational Operators ---------- /

* Operator *  * Result *
```

==	equal to
!=	not equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

/* ----------------- Boolean Operators ----------------- /

* Operator *	* Result *
&	logical AND
|	logical OR
^	logical XOR
||	short-circuit OR
&&	short-circuit AND
!	logical NOT

// Short-circuit AND, &&, only checks the second condition if the first is true.
// Short-circuit OR,||, only checks the second condition if the first is false.

/* ---------------- Operator Precedence ---------------- /

* Operator *	* Associativity *

Highest	()	[]	.			left to right
		++	--	!		right to left
		*	/	%		left to right
		+	-			left to right
		>	>=	<	<=	left to right
		==	!=			left to right
		&				left to right
		^				left to right
		|				left to right
		&&				left to right
		||				left to right
Lowest	=					right to left

When there are two operators with the same precedence the expression is evaluated according to its associativity.

/ ---------- */

```
		/********************** 1.0 - Ask user for a valid full name **********************/


		// initialize Scanner class to getting input from user
		Scanner scan = new Scanner(System.in); // create an instance of the Scanner class named as scan




		// Loops allow your program to do the same thing again and again and again and again.
		// The following loop will ask the user for thier name until they enter a valid name.
		// the variable validName is currently false. We will set it to true when we get a valid name.
		while(!validName)
		{
			// this loop that will run until the user enters a valid name.


			// ask user to enter their name
			System.out.println("What's your full name?");


			// scan the next line that the user enters and store it in the variable 'name'.
			name = scan.nextLine();
```

```
                        /* other important scanner methods:
                                scan.nextInt(); // reads in the next token as an
int
                                scan.nextFloat(); // reads in the next token as
an float
                                scan.nextDouble(); // reads in the next token
as an double
                                scan.nextLine(); // reads in the next token as
an String
                                scan.hasNextInt(); //returns true if there is
another int to read in
                                scan.hasNextFloat(); //returns true if there is
another float to read in
                                scan.hasNextDouble(); //returns true if there is
another double to read in
                                scan.hasNextLine(); //returns true if there is
another String to read in
                        */



                        // check if the user has entered their full name
                        // we are checking where the space character appears in the
name
                        // we will get -1 if the character does not exist
                        // use == to compare
                        if(name.indexOf(' ') == -1)
                        {


                                // there is no space in the name, so user has not
entered their full name
```

```
                    validName = false; // it's not necessary to set this to
false here again since we have not changed it since initializing it


                    System.out.print("Oops! Looks like you forgot to enter
your full name... ");


                }
                else
                {
                    // there is a space in the name, so the name must be
valid
                    validName = true;
                    // setting this variable to true will exit out of the while
loop
                }


            }


/**********************************************************************
*************/
```

```
                /***************** 1.1 - Split the full name to first and last name
*****************/


                // indexOf() Returns the index within the string of the first occurrence
of the specified character or -1 if the character does not occur
                int spaceIndex = name.indexOf(' ');


                // Strings are indexed starting from 0
                // get the characters from 0 to where the space is
                firstName = name.substring(0, spaceIndex);


                // consider everything after the first space as the last name
                // get the characters one place after space to the end
                lastName = name.substring(spaceIndex + 1);
```

```
/**********************************************************************
*************/
```

```
                /****************** 1.2 - Capitalize the first char of the names
*******************/

                // lowercase all the char in name
                name = name.toLowerCase();
```

```
                // capitalize the first letter of the names
                // get the first char and change that to upper case and then get the rest of the characters after the 1st char
                firstName = firstName.substring(0, 1).toUpperCase() + firstName.substring(1); // .toUpperCase() converts a String to uppercase chars
                lastName = lastName.substring(0, 1).toUpperCase() + lastName.substring(1);


                // greet the user with their first name...
                System.out.println("\nHello there " + firstName + "! I hope to help you revise some Java...");




                // ----- let's find some interesting things about the name...


                // \n prints a new line
                System.out.println("\nHere are some interesting facts about your name:");


                System.out.println("* Your first name: " + firstName);
                System.out.println("* Your last name: " + lastName);



/*****************************************************************************/
```

```
                /********************** 1.3 - Find the length of the name
**********************/


                // check how long the string is
                // .length() gives the length of a String as an <int>
                System.out.println("* Your name is " + name.length() + " characters
long.");
```

```
                /********************** 1.4. Check if firstName == lastName
**********************/


                // check if the first name is the same as the last name

                // short hand if statement without the {} brackets

                if(firstName.equals(lastName)) System.out.println("* Your first name
and last name is the same!");



/*****************************************************************************
*************/
```

```
        /****************** 1.5 - Find how many vowels are in the name
********************/


        // create an array with all the vowels.
        // An array is a collection of variables.
        char vowels[] = new char[5]; // declare and allocate memory for the
array
        vowels[0] = 'a'; // arrays are indexed from 0.
        vowels[1] = 'e';
        vowels[2] = 'i';
        vowels[3] = 'o';
        vowels[4] = 'u';



        // declare and initialise together
        char vowelsCaps[] = {'A', 'E', 'I', 'O', 'U'};


        /* Arrays have three important properties:
                - represent a group of related data.
                - all data within an array have the same type.
                - size of an array is fixed once it is created.
        */
```

```
                int vowelsCount = 0; // counter that will keep track of the number of vowels

                // create a new string and store the name into this
                String highlightedName = name;

                // for loop: (initialisation; condition; update)
                for(int letter = 0; letter < name.length(); letter++) // outer loop
                {

                        // this outer loop will run for the length of the name



                        // to get the length of an array, use .length - Note that there are no () in the end.
                        for(int i = 0; i < vowels.length; i++) // inner loop
                        {

                                // this inner loop will run for the length of the array (5 times) everytime the outer loop runs.

                                // check if char at outer loop count (letter) is equal to the char in vowels array at position of the inner loop count
                                if(name.charAt(letter) == vowels[i])
                                {
```

```
                                        // increase the count of vowels we have in the
name
                                        ++vowelsCount; // Increment variable



                                        // capitalize the vowels to highlight them
                                        // replace lowercase vowels in the name with
uppercase vowels
                                        // .replace() Replaces the first char specified
with the second character specified
                                        highlightedName =
highlightedName.replace(vowels[i], vowelsCaps[i]);


                                    }


                                } // END of inner loop


                            } // END of outer loop


                            System.out.println("* Your name has " + vowelsCount + " vowels (" +
highlightedName + ").");


/**********************************************************************
**********/
```

```
		/********************** 1.6 - Check if name is palindromic **********************/

		// palindrome: a word, phrase, or sequence that reads the same backwards as forwards.

		// assume that the name is palindromic - we will set this to false if we find that it is not
		boolean palindrome = true;

		// convert all char to the same case as A is not equal to a.
		String nameUpper = firstName.toUpperCase(); // .toUpperCase() convert a String to uppercase chars
```

```
                // loop through the name
                for(int count = 0; count < nameUpper.length(); count++)
                {
                        // check if the char at count is the same as the char at the
other end
                        if(nameUpper.charAt(count) !=
nameUpper.charAt(nameUpper.length() - 1 - count))
                        {
                                // if the corresponding char is not the same, set
boolean to false
                                palindrome = false;


                                // break out of the loop once we have found a char
that is not palindromic
                                break;
                        }
                }


                if(palindrome)
                {
                        System.out.println("* Your first name is palindromic (reads the
same backwards as forwards)!");
                }
                else
                {
                        System.out.println("* Your first name is not palindromic.");
                }



/*****************************************************************
*************/
```

```
                    /****************** 2.0 - Ask user for a valid month and year
*******************/
```

```
        // while loop to keep asking user for a valid month
        while(month == 0)
        {
            // ask for name of the month
            System.out.print("\nName the month were you born in: ");
            String monthName = scan.nextLine();



            // ---------- get the number of the month


            // convert user input to lowercase so we can match uppercase and lowercase input
            monthName = monthName.toLowerCase(); // convert String to lowercase chars


            // A switch statement gives us the option to test for a range of values for our variables.
            // They can be used instead of long, complex if ... else if statements.
            switch(monthName)
            {
                case "january": month = 1; break;
                case "february": month = 2; break;
                case "march": month = 3; break;
                case "april": month = 4; break;
                case "may": month = 5; break;
                case "june": month = 6; break;
                case "july": month = 7; break;
                case "august": month = 8; break;
                case "september": month = 9; break;
```

```
                    case "october": month = 10; break;
                    case "november": month = 11; break;
                    case "december": month = 12; break;
                    default: month = 0; break; // if we cannot find a valid month
                }


                // print out error message if not a valid month
                if(month == 0) System.out.println("Emmm... that doesn't look like a valid month...");
            }


            // ------------------------ [ Ask for a valid birth year ] ------------------------ //

            int count = 0; // control variable to keep track of how many times the loop has run

            // this is a do-while loop. it will run at least once and then check the condition at the bottom to determine if it needs to run again
            do
            {
```

```
                        // if we are in the loop again...
                        if(count > 0){
                                System.out.println("\nEmmm... That doesn't look like a
year you could be born in! ");
                                System.out.println("Try a year between 1930 and
2010... ");
                        }


                        // ask for the birth year
                        System.out.println("What year were you born in?");
                        year = scan.nextInt();


                        // check if this is a valid year. must be born on or after 1930
and before 2011.
                        validYear = (year >= 1930 && year < 2011) ? true : false; //
The ternary operator
                        // if condition is true, set boolean variable validYear to true or
else false.
                        /* The ternary operator takes three arguments:
                                - a condition, a true value and a false value.
                                - It tests the condition and then returns one of two
values to the variable based on the result of the condition.
                        */


                        // keep track of how many times we are in the loop
                        count++; // update


                        /* ----- difference between i++ and ++i
```

++i will increment the value of i, and then return the incremented value:

i = 1;

j = ++i;

(i is 2, j is 2)

i++ will increment the value of i, but return the original value that i held before being incremented:

i = 1;

j = i++;

(i is 2, j is 1)

---------- */

} while(!validYear); // condition

/**********
**********/

```
        /************************ 2.1 - Calculate how old you are
************************/


        // ---------- Find some more interesting facts...
        System.out.println("\nSome intersting facts about your birthday:");




        // print out which month they were born in
        if(month == 1)
        {
                System.out.println("* You were born on the 1st month.");
        }
        else if(month == 2)
        {
                System.out.println("* You were born on the 2nd month.");
```

```
                }
                else if(month == 3)
                {
                        System.out.println("* You were born on the 3rd month.");
                }
                else
                {
                        System.out.println("* You were born on the " + month + "th
month.");
                }



                // calculate their age
                age = 2013 - year;
                System.out.println("* You are " + age + " years old.");


                // check how old they are in different units :P
                System.out.println("* You are " + (age * 12) + " months old.");
                System.out.println("* You are around " + (age * 52) + " weeks old.");
                System.out.println("* You are approximately " + (age * 365) + " days
old.");
                System.out.println("* You are over " + (age * 356 * 24) + " hours
old.");


/**********************************************************************
*************/
```

```
		/************************** 2.2 - Odd or Even month?
**************************/


		// check if the age is an odd or even number
		if(age % 2 == 0)
		{
			System.out.println("* Your age is an even number.");
		}
		else
```

```
                {
                        System.out.println("* Your age is an odd number.");
                }


                /********************** 2.3 - Were you born on a leap year?
**********************/


                // check if the year is divisible by 400 OR (is divisible by 4 AND is NOT
divisible by 100)
                if((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0)))
                {
                        System.out.println("* The year " + year + " is a leap year!");
                }
                else
                {
                        System.out.println("* You were not born on a leap year.");
                }



/*************************************************************************
*************/
```

```
                /****************************** 2.4 - Lucky number
*******************************/

                // initialize Random class to generate random number
                Random randomNumber = new Random(); // create an instance of the
Random class named as randomNumber

                // get a random number between 1 to 12
                int luckyNumber = randomNumber.nextInt(12);

                // get the digits in the ones, tens, hundreds and thousands place of
the birth year
                // % means modulus (mod) in Java and it calculates the remainder
after division
                // e.g. if the year = 2013
                int onesDigit = year % 10; // onesDigit = 3
```

```
                int tensDigit = year / 10 % 10; // tensDigit = 1
                int hundredsDigit = year / 100 % 10; // hundredsDigit = 0
                int thousandsDigit = year / 1000 % 10; // thousandsDigit = 2

                // raise number to the power of 4
                double powerDigit = Math.pow(thousandsDigit, 4);

                // add and subtract some digits from the birth year
                int result = onesDigit - hundredsDigit + tensDigit;

                // add result with the randomly generated number
                luckyNumber += result; // Addition assignment. Same as:
luckyNumber = luckyNumber + result;

                System.out.println("* Your lucky number is: " + luckyNumber);

/**********************************************************************
*************/
```

```
		/****************************** 2.4 - Lucky Dates ******************************/

		// create an int array of length 3
		int luckyDates[] = new int[3];

		// fill array with random numbers
		for(int i = 0; i < luckyDates.length; i++)
		{
			// generate a random number between 0 and 30 and add 1 so we dont get 0
			int randomDate = randomNumber.nextInt(30) + 1;
```

```
            // store the randomly generated number to array
            luckyDates[i] = randomDate;
        }


        System.out.print("* Your lucky dates are: ");


        // loop through the array again and get the values stored
        for(int i = 0; i < luckyDates.length; i++)
        {
            // if we are on the last array position
            if(i == luckyDates.length - 1)
            {
                // print '&' before and full stop in the end.
                System.out.print("& " + luckyDates[i] + ". \n\n");
            }
            else
            {
                // print number with ',' in the end
                System.out.print(luckyDates[i] + ", ");
            }
        }



/**********************************************************************
*************/
```

```
        } // END main

} // public class
```

```
/*

-----[ Some Programming Theory ]-----

* A computer program is a collection of instructions that describes a task, or set of tasks, to be carried out by a computer.

* A quality program is...
```

* Readable

- Other programmers should be able to easily read and understand what your code does.
- You should indent your code.
- Add comments to explain what the program does.

* Modular

- Programs should be broken down into component parts, where each part is subdivided as necessary.

* Robust

- A program should gracefully handle cases when the input is not as expected or some other error has occurred.
- A program should never crash.

-----[Things you need to know about Java]-----

* Java is a programming language. It can be used to write applications and applets.
* It is an object-oriented language.
* Platform-independent because of the JVM (Java Virtual Machine).

- "Write once, run anywhere"

-----[Writing a Java Program]-----

* Written in plain text format and saved with a .java file extension (like this file).
* But, the machine won't understand this.
* So, we need to compile the program to turn it into Java bytecode.

- The bytecode is stored in .class file (this file will be created if you build this java program).
- Bytecode is a highly optimised set of instructions designed to be executed by a JVM (Java Virtual Machine).

* A JVM interprets the bytecode and runs it on the machine.
* The compiler is called javac and the JVM interpreter is called java.

* JVM = Java Virtual Machine

* JVM is just a software program that allows the same .java files to run on many machines.

* Java is portable because it relies on a layer of Software and Hardware. Each layer only interacts with neighbouring layers.

- Your Java Program <-> JVM <-> Operating System/Embedded Software <-> Hardware of PC/Mobile Phone

* Java is case sensitive.

* All statements in Java end with a semi-colon.

-----[Steps to writing a program]-----

1. Develop an algorithm.
2. Write a software implementation of the algorithm - a software program.
3. Compile it.
4. Fix any compilation errors.
5. Test it - try to run it.
6. Fix any runtime errors.

*/

/*

What does this program do?

1.0. Asks for your full name until you enter your full name. Then finds interesting things with your name.

- uses scanner, while loop, boolean variable, if statement.

1.1. Splits your first and last name to separate strings.

- uses for loop, strings, substrings, if statement.

1.2. Capitalize the first char of the names.

- uses strings, substrings and string functions - <string>.toUpperCase(); and <string>.toLowerCase();.

1.3. Finds how long the name is.

- uses string, <string>.length() function.

1.4. Check if your first name is equal to your last name.

- uses <string1>.equals(<string2>); function.

1.5. Finds how many vowels are in your name with a nested loops and arrays of vowels.

Then replaces the the vowels in your name with the same capital char to highlight them.

- uses char arrays, nested for loops, if statement, <string>.replace(<char>, <char>); function.

1.6. Checks if your first name is palindromic.

- uses a for loop, if statement and loop break;

2.0. Repeatedly asks for the month you were born in and for the year until you enter a valid year.

Match the month name to find the month's number.

- uses while loop, do-while loop, scanner, if statement, switch statement, string, int variables.

2.1. Calculates how old you are in different units.

- uses if, else if statements, multiplying numeric operator

2.2. Checks if your age is an odd or even number.

- uses modulus, if statement

2.3. Check if your birth year was a leap year.

- uses modulus, if statement

2.4. Gets each digit of your birth year and randomly finds your lucky number.

- uses random class, modulus to find a single digit, double variables, numeric operators and math.pow(<x>,<y>) function.

2.5. Randomly fills an array with numbers from 1 to 31. These display as your lucky dates.

- uses array, for loop, if else statements.

----- That just about covers everything we have learned in programming so far... -----

*/