Object-Oriented Programming Software Workshop 1

Chapter 2

What is meant by a constant and a variable?

- **Constant**
  - A data item that cannot be changed
- Literal constant
  - Value taken literally at each use
- Numeric constant
  - As opposed to a literal constant
- Unnamed constant -No identifier is associated with it
- **Variable**
  - A named memory location
  - Used to store a value
  - Can hold only one value at a time
  - Its value can change

Data Types

- A type of data that can be stored
- How much memory an item occupies
- What types of operations can be performed on data

Primitive

- A simple data type

Reference types

- More complex data types
- 8 primitive data types

Keyword	Description
byte	Byte-length integer
short	Short integer
int	Integer
long	Long integer
float	Single-precision floating point
double	Double-precision floating point
char	A single character
boolean	A Boolean value (ture or false)

- Integer Data Types

Type	Minimum Value	Maximum Value	Size in Bytes
byte	-128	127	1
short	-32,768	32,767	2
int	-2,147,483,648	2,147,483,647	4

Type	Minimum Value	Maximum Value	Size in Bytes
long	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	8

- boolean Data Type

Operator	Description
<	Less than
>	Greater than
==	Equal to
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to

- Floating-Point Data Types

Type	Minimum	Maximum	Size in Bytes
float	$-3.4 * 10^{38}$	$3.4 * 10^{38}$	4
double	$-1.7 * 10^{308}$	$1.7 * 10^{308}$	8

- Char Data Type

Escape Sequence	Description
\b	Backspace;moves the cursor one space to the left
\t	Tab; moves the cursor to the next tab stop
\n	Newline or linefeed; moves the cursor to the beginning of the next line
\r	Carriage return; moves the cursor to the beginning of the current line
\"	Double quotation mark; displays a double quotation mark
\'	Single quotation mark; display a single quotation mark
\\\\	Backslash; displays a backslash character

Declaring variables

- Provide date type & Name
- For example
  - int myNumber;
- Include a starting value
  - int myNumber = 25
- = is the assignment operator
  - Assigning the value 25 to memory location

Declaring constants

- Use upper case
- Use the keyword final
- For example
  - final double TAX_RATE = 500.95;
  - payAmount = hoursWorked * TAX_RATE;

Reference types

- Holds a memory address
- Class name
- String surname = "Smith"

Accepting input from the user

- 2 ways of achieving this for now
    - Command prompt
    - Using a GUI environment
- Command prompt
    - Scanner class
- GUI
    - JOptionPane class

Scanner class

Method	Description
nextDouble()	Retrieves input as a double
nextInt()	Retrieves input as an int
nextLine()	Retrieves the next line of data and returns it as a String
next()	Retrieves the next complete token as a String
nextShort()	Retrieves input as a short
nextByte()	Retrieves input as a byte
nextFloat()	Retrieves input as a float. Note that when you enter an input value that will be stored as a float, you do not type an F. The F is used only with constants coded within a program.
nextLong()	Retrieves input as a long. Note that when you enter an input value that will be stored as a long, you do not type anL. The L is used only with constants coded within a program.

- Example

```
import java.util.Scanner;
public class GetUserInfo
{
    public static void main(String[] args)
    {
        int num;
        Scanner userInput = new Scanner(System.in);
        System.out.println("Please enter a value for the number>>");
        num = userInput.nextInt();
    }
}
```

```
import java.util.Scanner;

public class LetsInputScanner{
    public void inputPrint(){
        Scanner sc = new Scanner(System.in);

        int mark = 0;
        int studId;
        String name;
        System.out.printf("Please enter your student id %n>>");
        studId = sc.nextInt();
        sc.nextLine();
        System.out.printf("Please enter your name %n>>");
        name = sc.nextLine();
        System.out.printf("Please enter the mark %n>>");
        mark = sc.nextInt();
        System.out.printf("Student %s with student ID %d has %d", name, studId, mark);
    }

    public static void main(String[] args){
        LetsInputScanner inputting = new LetsInputScanner();
        inputting.inputPrint();
    }
}
```

Using the JOptionPane class for input

```
import javax.swing.JOptionPane;
public class InputJOP{
    public static void main(String[] args){
        String theAnswer;
        int answer;
        theAnswer = JOptionPane.showInputDialog(null, "Please enter a number");
        answer = InputJOP.parseInt(theAnswer);
        JOptionPane.showMessageDialog(null, " You typed " +theAnswer);
        JOptionPane.showMessageDialog(null, " You typed in " +answer +" You typed theAnswer " +theA
    }
}
```

Looking back the Chapter 2

- Declaring and using constants and variables
- Learning about the different data types
- Using the Scanner class to accept keyboard input
- Using the JOptionPane class to accept GUI input
- Using calculations within a Java class
- Understanding Type conversion

Chapter 5

Planning Decision-Making Logic

- Pseudocode
    - Use paper, and a pencil
    - Plan a program's logic by writing plain English statements
    - Accomplish important steps in a given task
    - Use everyday language
- Flowchart
    - Steps in diagram form
    - A series of shapes connected by arrows
- Sequence structure
    - One-step follows another unconditionally
    - Cannot branch away or skip a step
- Decision structure
    - Involves choosing among alternative courses of acting
    - Based on some value within a program
- Boolean values
    - true and false values
    - Used in every computer decision

The If...else Statement

```
import java.util.Scanner;

public class IfElse{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.printf("Please choose 1 - queen, 2 - king, 3 - pullout%n");
        int Value = sc.nextInt();
        sc.nextLine();
        if (Value == 1){
            System.out.printf("You have chosen queen for £120%n");
        }else if(Value == 2){
            System.out.printf("You have chosen king for £139%n");
        }else{
            System.out.printf("You have chosen pullout for £190%n");
        }
    }
}
```

The Switch Statement

```
import java.util.Scanner;

public class Switch_{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        System.out.printf("Please choose 1 - queen, 2 - king, 3 - pullout%n");
        int Value = sc.nextInt();
        sc.nextLine();
        switch (Value) {
            case 1:
                System.out.printf("You have chosen queen for £120%n");
                break;
            case 2:
                System.out.printf("You have chosen king for £139%n");
                break;
            case 3:
                System.out.printf("You have chosen pullout for £190%n");
                break;
        }
    }
}
```

Operator Precedence

Precedence	Operator(s)	Symbol(s)
Highest	Logical NOT	!
Intermediate	Multiplication, division, modulus Addition,subtraction Relational Equality Logical AND Logical OR Conditional	*/% +- ><>=<= == != && || ?:
Lowest	Assignment	=

Chapter 7

String

- Class
- Sequence of characters
- Examples of strings
  - System.out.println("This is a string of characters");
  - "John Smit"
- String myString = "This is my string";
- String otherString = new String("This is another string");

Comparing strings

- Understanding how strings are stored in memory
- String name1 = "John";
- Strings are immutable

Comparisons between strings

```
public class CompareString{
    public static void main(String[] args){
        String name1 = "John";
        String name2 = "John";
        if (name1 == name2){}
        if (name1.equal(name2)){}
    }
}
```

Other useful methods

Method	Description
isUpperCase()	Tests if character is uppercase
toUpperCase()	Returns the uppercase equivalent of the argument; no change is made if the argument is not a lowercase letter
islowerCase()	Tests if character is lowercase
tolowerCase()	Returns the lowercase equivalent of the argument; no change is made if the argument is not an uppercase letter
isDigit()	Returns true if the argument is a digit (0- 9) and false otherwise
isletter()	Returns true if the argument is a letter and false otherwise
isletterOrDigit()	Returns true if the argument is a letter or digit and false otherwise
isWhitespace()	Returns true if the argument is whitespace and false otherwise; this includes the space, tab, newline, carriage return, and form feed

- length()
  - Determine the length of a string
- Starting position? John 0123
- int sl = name1.length();
- name1.indexOf('J');
- int ind = name1.indexOf('J');
- System.out,println("The index of J is " + name1,indexOf('J'));
- char theChar = name1.charAt(3);
- What would that character be for the name John?
  - n
- System.out.println("The first character in the name is " + name1.charAt(0));
- toUpper()/toLower()
- name1 = name1.toUpperCase();

Summary

- String variables
  - References
- Character class
  - Instances can hold a single character value
- Each String class object
  - Is immutable
  - equals() method
- toString() method

    - Convert any object to a String
- Integer.parseInt() method
    - Takes a String argument and returns an integer value
- Double.parseDouble() method
    - Takes a String argument and returns a double value
- StringBuilder or StringBuffer class
    - improves performance when a string's contents must change