ICS108 NOTES

Airbus5717

June 12, 2021

Contents

THIS IS NOT AN ALTERNATIVE TO THE BOOK

1 Chapter 1: Introduction to Java

1.1 Simple Java Program

```java
class App {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Java source files are compiled by Java compiler to bytecode (.class files) then ran with Java Virtual Machine (JVM)

- Class name = App

- Main method = public static void main (arguments)

  { code in method }

- Statements = i.e. print statement

  - each statment in java must end with a semicolon (;)

- Reserved keywords

  - class
  - public
  - static
  - void
  - etc.

- Comments

  - single line and multiline comments

    ```java
    // single line comment
    /* multi line
     comment */
    ```

- Blocks: a group of components of a program

1.2 Programming Style and documentation

- Appropriate Comments

- Naming Convertions

- Proper Identation and spacing lines

- Block styles

1.3 Programming Errors

- Syntax Errors

  - Detected by the compiler (i.e. missing semicolon)

- Runtime Errors

  - Causes the program to abort (i.e. divition by zero)

- Logic Errors

  - Produces incorrect results (i.e. incorrect logic)

---

2 Chapter 2 : Elementry Programming

2.1 Program example:

```java
public class App {
    public static void main(String[] args) {
        double radius;
        double area;

        // assign a radius
        radius = 20;

        // Compute Area
        area = Math.pow(radius, 2) * Math.PI;
        // NOTE: Math pow function returns a double

        // Display result
        System.out.println("The Area: " + area + " for radius: " + radius);
```

```java
    }
}
```

2.2 Reading Input

Reading Input can be done by creating a Scanner Object which can be
imported from 'java.util.Scanner;'

```java
// import module.class
import java.util.Scanner;

class App {
    public static void main(String[] args ) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a double value: ");
        double d = input.nextDouble();

        // to get int: use 'input.nextInt();'
        // float: use input.nextFloat();
        // for String: use input.next(); or input.nextLine();
        // for Char use String input then
        // get the first char (code below)
        // String.s = input.nextLine();
        // char ch = s.charAt(0);

        input.close();
        /*
          good practice is to
          close scanners and files
          */
        // Display output
        System.out.println("the double value is " + d);
    }
}
```

2.3 Imports

- Implicit import (import java.util.*;)

- Explicit import (import java.util.Scanner;)

No Performance difference

2.4 Identifiers

- sequence of chars are from letters, digits, underscores(_) and dollar signs($).

- An identifier must start with a letter, an underscore or a dollar sign,

<div align="center">

IT CANNOT START WITH A DIGIT.

</div>

- An identifier cannot be a reserved word or default types such as (true, false etc.).

- An identifier can be of any length.

2.5 Variables

2.5.1 declare variables

```java
int x = 1; // variable example

// other variables
double y = 12.0;
char b = 's';
String u = "Bruh";
```

'int' is a type,
'x' is an identifier,
'1' is an int value,
';' is for statement termination,
'=' is for assignment

2.5.2 Constant variables

```
final int SIZE = 3;
// final keyword is written before datatype
// to indicate that the variable is immutable
```

2.6 Naming Conventions

choose meaningful names

2.6.1 Variable and method names

use lowercase and capitalize each word after the first word

```
int computeArea(int area, int radius) {
    int computedResult = area * radius; // example
    return computedResult;
}
```

2.6.2 Class names

Capitalize first letter of each word in the name for example

```
class ComputeArea {  }
```

2.6.3 Constant names

Capitalize all letters
    for example

```
final int MAX_VALUE = 100;
```

2.7 Operators

- (+) add

- (-) substract

- (*) multiply

- (/) divide

- (%) remainder i.e. 5 % 2 == 1

2.7.1 useful operations

```
i = i + 1;
i += 1;
i++;
++i;
 // these 4 statements are the same

++i; // adds then uses the value
i++; // uses the value then adds
--i;
i--;
// but if it is a statement by it self then
// it wouldn't matter much

// other operators's support
i += 1; i -= 1; i *= 1; i /= 1; i %= 1;
```

2.8 Data types

2.8.1 Integers

are numbers without decimal values and range between -2^{31} to $(2^{31}) - 1$
example:

```
final int MAX_INT =  2147483647;
final int MIN_INT = -2147483648;
```

```java
// example
int x = 100;
```

2.8.2   Floats and Double

are numbers with decimal points by default Java will make any decimal point double unless added an F after it i.e. letter D can be used for classifing as double.

```java
float x = 10.0f; // f is written to indicate that the variable is float
double y = 10.0;
// also correct
double y2 = 10.0d;
```

> NOTE: floating points are not accurate always during calculations and it is recommeneded to use double for more accuracy

---

2.8.3   Scientific Notation

Floating point literals can be specified in scientific notations using (e, E).

> NOTE: use double for more accuracy

for example

```java
double sciX = 10.2e20;
```

2.8.4   Chars and Strings

are used to store text, char are for one character and strings are used for multiple characters

```java
char b = 'a';
String str = "bruh why String is capital";
// NOTE: String data type first letter is capital
```

2.8.5   other types

- byte: similar to int but smaller range (-128 to 127)

- long: similar to int but bigger range (-2^{63} to (2^{63}) - 1)

2.8.6   Display Current Time in GMT

```
long time = System.currentTimeMillis();
// == current GMT time in milliseconds
```

2.8.7   Conversion rules

1. if one of the operands is double then final value is Double

2. otherwise if one is float then the final value is float.

3. otherwise, if one of the operands is long then both are long.

4. finally they are int if one of them is int

2.8.8   Type casting

- implicit casting i.e.

  ```
  double d = 3; // (type widening)
  ```

- Explicit casting i.e.

  ```
  int i = (int) 3.0; // (type narrowing)
  int j = (int) 3.9; // (fraction part is truncated)
  // i = 3; j = 3;
  ```

  another example

```
int sum = 0;
sum += 4.5; // now sum is 4
```

2.9   Common Errors and pitfalls

1. Common Errors

   (a) Undeclared Variables and unused variables i.e. using Variables that do not exist.

   (b) Interger overflow using numbers over the max/min range

   (c) Round-off Errors when dealing with alot of float numbers

   (d) Unintended Integer division i.e. division over zero

   (e) Redundant Input objects i.e. getting wrong input for example: getting a string instead of an int.

---

3   Chapter 3: Selections

3.1   More Data types

3.1.1   boolean type

bool values are true or false

```
boolean type = true;
type = false; // changed to false
```

3.1.2   boolean (comparasion operators)

$>$, $<$, $<=$, $>=$, etc. i.e.

```
bool x = 3 > 2; // true
bool y = 4 < x; // false
```

3.2   If else statements

3.2.1   if

checks for true boolean then excutes code in the block

3.2.2 else

if the 'if' condition is false then else block executes

```
int x = 1;

if (x > 0) {
    // if x is positive then this code block executes
    // NOTE: in this example the code here executes.
} else {
    // if x is negative the code here executes.
}


// also this is possible

if (x > 0) {
    // if x is positive
} else if (x < 0) {
    // if x is negative
} else {
    // if x is not positive nor negative
}

// the code will check at each statement
// also adding a semicolon at if or else is an error
// and it is a logic error
```

3.3 Logical operators

operator	name	description
!	not	logical negation
&&	and	logical conjunction
^	exclusive or	logical exclusive
||	or	logical disjunction

examples:

```
int x = 1;
if (x != 1) {
```

```
        // wont execute
}

int y = 0;
if (x == 1 && y == 0) {
        // will execute
}


// '^' operator
// if both are true or false then it will evaluate as
// false otherwise if one is false and the other isnt
// it will evaluate as true

//   false     true
if (x != 1 ^ y == 0) {
        // will execute
}

bool a = false;

if (!a)
        // will be true and execute

bool b = true;

if ( a || b ) {
        // will execute (true)
}
```

Leap year example

```
int year = 2021; // use input or get the year number

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        System.out.println("Year is leap");
}
```

3.4   Switch statement

alternative to if statements equating with specific value.

```
int x = 1;

switch (x) {
    case 1:
        // code if x == 1
        break;
    case 2:
        // code if x == 2
        break;
    default:
        // insert if value doesnt match the cases
        break;
}
```

3.5   Conditional operators: Ternary

(boolean) ? (if-true) : (else);

```
boolean x = true;

int b = x ? 1 : 0; // now b is 1 cuz x is true

int c = !x ? 1 : 0; // c is 0
```

3.6   Operator order (precedence)

1. var++, var−

2. +, -, and ++var, −var.

3. (type) cast

4. ! (Not)

5. *, /, %

6. +, -

7. <, <=, >, >=

8. == , !=

9. ˆ (Exclusive or)

10. &&

11. ||

12. ==, +=, -=, *=, /=, %=

3.7 Debugging

NOTE: use a debugger when facing problems

4 Chapter 4: Math functions, Chars and Strings

4.1 Math Class

4.1.1 Constants

- PI
- E

4.1.2 Methods

1. Trigonometric Methods return double always

    - sin(double a)
    - cos(double a)
    - tan(double a)
    - acos(double a)
    - asin(double a)

- atan(double a)
- toRadians(double a) // converts to radians

---

2. Rounding Methods

- ceil(double x) x is rounded up to nearest int then returned as double
- floor(double x) x is rounded down to nearest int then returned as double
- rint(double x) x is returned to the nearest int, if x is equally close to both then the even one is returned as double
- round(float x) returns (int)Math.floor(x + 0.5);
- round(double x) returns (long)Math.floor(x + 0.5);

    NOTE: check examples in slides for Chapter4 page:(8)

---

3. Exponent Methods

   (a) **TODO** Methods

---

4. min, max, abs, and Random Methods

- max(a, b) and min(a, b) return max or min of the 2 arguments
- abs(a) returns the absolute value
- random() returns a random double (from 0.0 to 1.0)

4.2   Characters

4.2.1   Special Chars

- \b Backspace

- \t Tab

- \n linefeed

- \f formfeed

- \r carriage return

- \\ Backslash = \

- " or \" Double Quote

4.2.2   Casting between char and ints

```
int i = 'a'; // same as int i = (int)'a';

char a = 97; // same as char c = (char)97;

char ch = 'a';// choose any value

if (ch >= 'A' && ch <= 'Z')
    // uppercase
else if (ch >= 'a' && ch <= 'z')
    // lowercase
else if (ch >= '0' && ch <= '9')
    // number character
```

4.2.3   Character Methods

- isLetter()

- isDigit()

- isWhitespace()

- isUpperCase()

- isLowerCase()

- toUpperCase()

- toLowerCase()

- toString()

4.3 Strings

4.3.1 String Length

```
String message = "welcome to java";
int length = message.length(); // String length
```

Strings are arrays of characters that start at 0

4.3.2 specific char from string

```
message.charAt(0); // is 'W'
```

4.3.3 Converting strings

```
"Welcome".toLowerCase(); // returns "welcome"
"Welcome".toUpperCase(); // returns "WELCOME"
"Welcome  ".trim(); // returns "Welcome"
```

4.3.4 String concatenation

```
String s3 = s1.concat(s2);
// or
String s4 = s1 + s2;
// s3 == s4 will return true
```

4.3.5 String Methods list

Method	Description
charAt()	Returns the character at the specified index (position)
codePointAt()	Returns the Unicode of the character at the specified index
codePointBefore()	Returns the Unicode of the character before the specified index
codePointCount()	Returns the Unicode in the specified text range of this String
compareTo()	Compares two strings lexicographically
compareToIgnoreCase()	Compares two strings lexicographically, ignoring case differences
concat()	Appends a string to the end of another string
contains()	Checks whether a string contains a sequence of characters
contentEquals()	Checks whether a string contains the exact same sequence of characters of
copyValueOf()	Returns a String that represents the characters of the character array
endsWith()	Checks whether a string ends with the specified character(s)
equals()	Compares two strings. Returns true if the strings are equal, and false if no
equalsIgnoreCase()	Compares two strings, ignoring case considerations
format()	Returns a formatted string using the specified locale, format string, and ar
getBytes()	Encodes this String into a sequence of bytes using the named charset, stori
getChars()	Copies characters from a string to an array of chars
hashCode()	Returns the hash code of a string
indexOf()	Returns the position of the first found occurrence of specified characters in
intern()	Returns the canonical representation for the string object
isEmpty()	Checks whether a string is empty or not
lastIndexOf()	Returns the position of the last found occurrence of specified characters in
length()	Returns the length of a specified string
matches()	Searches a string for a match against a regular expression, and returns the
offsetByCodePoints()	Returns the index within this String that is offset from the given index by
regionMatches()	Tests if two string regions are equal
replace()	Searches a string for a specified value, and returns a new string where the
replaceFirst()	Replaces the first occurrence of a substring that matches the given regular
replaceAll()	Replaces each substring of this string that matches the given regular expre
split()	Splits a string into an array of substrings
startsWith()	Checks whether a string starts with specified characters
subSequence()	Returns a new character sequence that is a subsequence of this sequence
substring()	Extracts the characters from a string, beginning at a specified start positio
toCharArray()	Converts this string to a new character array
toLowerCase()	Converts a string to lower case letters
toString()	Returns the value of a String object
toUpperCase()	Converts a string to upper case letters
trim()	Removes whitespace from both ends of a string
valueOf()	Returns the string representation of the specified value

4.3.6 Obtaining substrings

- substring(int from, int to); `// to not included`

- indexOf(char ch); `gets index of char in string`

- substring(int from) `gets string from index till end of str`

4.3.7 Converting between Strings and Numbers

- Integer.parseInt(String s); `converts string to int`

- Double.parseDouble(String s); `converts str to double`

4.4 Formatting output

use the printf statement

- `%b` for boolean value

- `%c` for char value

- `%d` for integer

- `%f` for floating point number

- `%e` for std scientific notation

- `%s` for string

```
// System.out.printf(format, items);
String greet = "World!";

System.out.printf("Hello, %s", greet);
```