ICS108 NOTES

Airbus5717

June 9, 2021

Contents

THIS IS NOT AN ALTERNATIVE TO THE BOOK

---

1 Chapter 1: Introduction to Java

1.1 Simple Java Program

```
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

    ```
    // 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) * 3.14159;
        // NOTE: Math pow function returns a double

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

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();
        // etc
        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,

  IT CANNOT START WITH A DIGIT.

- 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

```
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;

// 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.

```
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

```
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

```java
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
ˆ	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

```java
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.

```java
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);

```java
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
