Logic Building Sessions - Class Notes

Java Basics

History of Java

- Most stable version – java 1.8
- Initial name – oak
- extentions - .java, .class..... .jar
- Developed by Sun Microsystems
- Taken over by oracle
- Java works on WORA principle – write once run anywhere

Parts of java

1. Core Java
2. Advanced Java
    a. j2se: java 2 standard edition
    b. j2ee: java 2 enterprise edition
    c. j2me: java 2 micro edition
    d. Java card edition

frameworks

1. spring
2. springboot
3. hibernate
4. jdbc
5. jsp

6. servlet

java features:

1. simple:
    - No pointers concept in java
    - No multiple inheritance in java
2. platform independent
    - .class file
3. Compiled & interpreted
4. Object oriented programming
5. Secure Language
6. Dynamic Language – runtime memory allocation
7. High performance
8. Robust – Memory management for garbage collection
9. Multithreaded
10. Portable & distributed
11. Architectural neutral

Dataypes and Variables

Variables

1. Local variables
2. Instance variables
3. Static Variables

Data Types

1. Primitive Datatypes

a. byte – 1 byte [-128, 127]
b. short – 2 bytes [-32,768, 32,767]
c. int – 4 bytes
d. long – 8 bytes
e. float – 4 bytes
f. double – 8 bytes
g. char – 2 bytes
h. boolean – 1 bit (true/false) (No 0 and 1 are allowed in java for boolean)

2. Non primitive datatypes
    1. String – Non primitive (Class in java)

Declaration of Variable with Datatype:

```
dataType Name;           //declare
dataType Name1 = 10;      //Initialise
int a =10, b=11;
```

→ Tokens = Smallest individual part of program

→ Identifier = names given to class, methods, variables

Rules for defining identifier–

1. Start with a character or '_' or '$' only
2. Numbers are not allowed at start
3. Duplicates are not allowed
4. Case sensitive (Camel-Casing is used for variable naming and Pascal-Casing for Class as a good practice)
5. Pre-Defined Keywords cannot be used as identifiers eg. int, float (Classnames can be used for variable names eg. String)

6. Constants should be written be in capital letters as a good programming practice (MAX_VALUE)

Conditional Statements

Java Statements

1. If

   Syntax:

   ```
   if(Condition){

   Statements;

   }
   ```

2. If-else

   Syntax:

   ```
   if(Condition){

   Statements;

   }else {

   Statements;

   }
   ```

3. If- else if - else

   Syntax:

```
if(Condition){

Statements;

}else if{

Statements;

}else {

Statements;

}
```

4. Switch

   Syntax

```
switch(expression)
{
  case value1 :
     Statements;
     break; // break is optional

  case value2 :
     Statements;
     break; // break is optional
     ....
     ....
   default :
     Statement; //it would run when no other cases got run
}
```

> 💡 Expressions with Double, float and long datatypes are not allowed in switch statements.

→ Switch statement can also have only default statement.
→ We cannot write a statement directly into switch block without using case.

5. Ternary Expression

Loops in Java

Loops

1. for

```
for (initialization; Condition; increment/decrement)
{
     // body of the loop
     statements;
}
```

The for loop would run in this sequence

> 💡 Initialization → Condition → Loop Body → Increment → Condition → Loop Body → Increment .... So on

A boolean value should be there in condition block.

▼ Snippets

```
for(int i=0, j=1 ;  i<5 ; i++,j++) {}; //Both a,b would
get initialized and incremented
```

```
for(;;) { } //infinite loop

int i = 0;
for(System.out.println("Init."); i<5; i++){
    System.out.println("Hello");
} //prints “init” for 1 time and “inside the loop” for
5 times

for(int i = 0; i<5; System.out.println("Increment")){
    System.out.println("Hello");
    i++;
} //prints hello, then increment for 5 times

for(int i=0; i<5;i++);
{
    System.out.println(“Hello”);
} //For loop would get terminated instantly and the blo
ck would get run separately
```

2. while

```
while (condition) {
    Statements;
}
```

For Loop	While Loop
The for loop is used when the number of iterations is known.	The while loop is used when the number of iterations is unknown.
Condition is a relational expression.	The condition may be an expression or non-zero value.
Initialization may be either in the loop statement or outside the loop.	Initialization is always outside the loop.
Once the statement(s) is executed then increment is done.	The increment can be done before or after the execution of the statement(s).

> 💡 We cannot directly feed false condition to while loop.

3. do-while

```
do{
    statements;
}while(condition);
```

Do while loop firstly runs the statements inside the block and then checks for the condition.

> 💡 Continue Statement: To skip an iteration
> Break Statement: To get out of the loop

Operators in Java

Operators in Java

1. arithmetic + , - , / , * , %
2. logical &&, ||, !
3. relational <, >, <=, >=, ==, !=
4. assignment =, +=, -=, *=, /=, %=
5. bitwise &, |, ^, ~
    - **&, Bitwise AND operator:** returns bit by bit AND of input values.
    - **|, Bitwise OR operator:** returns bit by bit OR of input values.
    - **^, Bitwise XOR operator:** returns bit-by-bit XOR of input values.

- **~, Bitwise Complement Operator:** This is a unary operator which returns the one's complement representation of the input value, i.e., with all bits inverted.

6. Shift Operators <<, >>

   → left shift operator:

   for 5<<2, it will add 2 bits to binary format of 5

   5 = 101

   (5<<2) = 101**00 = 20**

   → right shift operator:

   for 5>>2, it will remove last 2 bits of binary format of 5

   5 = 101

   (5>>2) = ~~101~~ **= 1**

7. unary +, -, ++, --

   → post: First use the value and then increment (a++)

   → pre: First increment then uses the value (++b)

8. ternary

```
variable = Expression1 ? Expression2: Expression3
                        //Check       For True     For Fal
se
```

Object Oriented Programming in Java

Class and Object

→ Class is a blueprint for creating objects

→ it has its own properties and behaviors.

→ Objects are instances of class which are user defined

> ClassName ObjectName = new ClassName();

here, **“new” keyword is used to allocate memory to object inside heap.**

```
Laptop obj = new Laptop(); //non-parameterised constructor

Laptop obj1 = new Laptop(2048,"Asus","i7","Windows",70000.5
f); //parametarised constructor
```

→ In java, Variables are: Local, Instance, Static

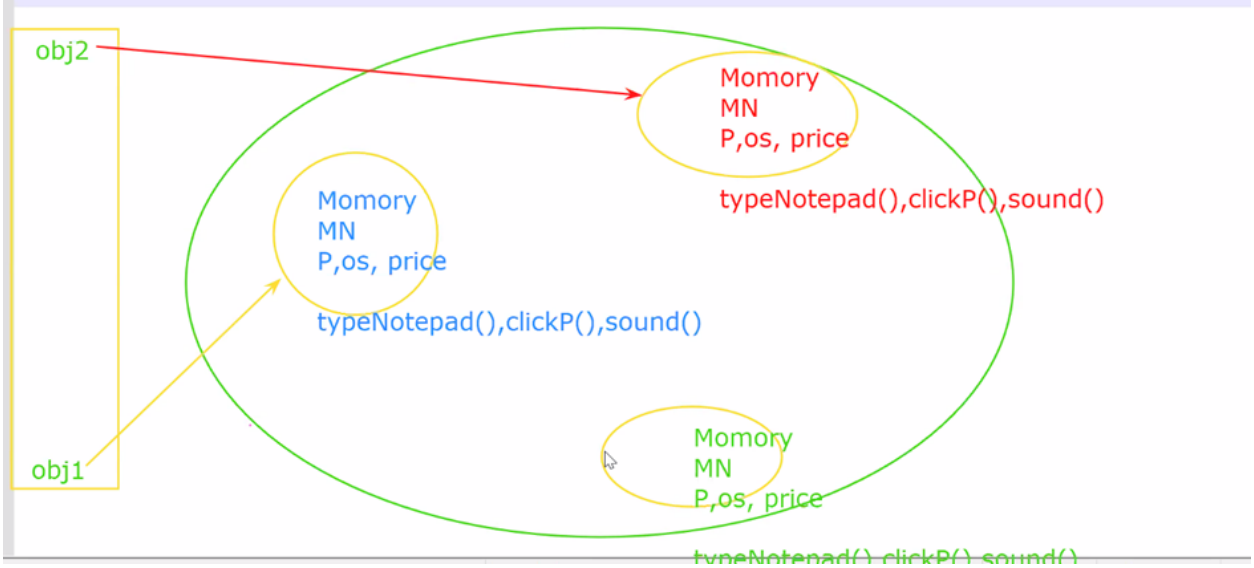
→ In instance variable of object, if we do not initialize them, they will get default value.

→ Classes & Objects are stored in heap memory of JVM.

→ Constructors are used to initialize the instance variables.

→ Constructors are methods without return type

→ When there is no constructor defined, default constructor is called. Default constructor sets default values to all instance variables



Objects and classes are stored in heap memory while the initializations and calls are made in stack memory

→ Static methods and variables are same for all objects and can be called by object and class itself.

→ Static methods and variables stored in heap for only once and called as required

→ Static variables can be accessed in instance methods as well as static methods

→ Instance variables cannot be accessed in static methods.

→ Static method once updated any instance, will remain changed for every instance.

```
class Student{
    int id;                 //Instance Variable
    String name;
    float marks;
    long mob;
    String address;
    static String instituteName = "CDAC Kharghar";     //Stat
ic Variable

    Student(){}             //Non parameterised constructor

    Student(int a, String b, float c, long d, String e){
//parameterised constructor
        id = a;
        name = b;
        marks = c;
        mob = d;
        instituteName = e;     //Static variable can be acces
sed through constructor
    }

    float displayMarks(){    //Instance method
        return marks;
    }
```

```
    static void calculatePercentage(){
        System.out.println("percentage");     //Static method
    }

    void displayDetails(){                  //Instance method
        System.out.println(id+" "+name+" "+marks+" "+mob+" "+
address+" ");
    }
}
```

→ this keyword is used to eliminate the confusion between class attributes and parameters with the same name.
→ ”this” keyword will point the current instance of class

→ init block - The initializer block contains the code that is always executed whenever an instance is created and it runs each time when an object of the class is created.

→ static block - The code inside the static block is executed only once: the first time the class is loaded into memory.

order of execution:

> Static block → init block → Constructor

```
class Student1{
    {
        System.out.println("Inside INIT");
    }

    static{

        System.out.println("Inside Static");
    }

    Student1(){
```

```
        System.out.println("Inside Constr.");
    }
```

→ Method overloading: methods which belongs to same class with same name and different parameters. (The two types of constructors shown in earlier codes was example of Method overloading)