2020

01

WK 14 (092-274)

WEDNESDAY

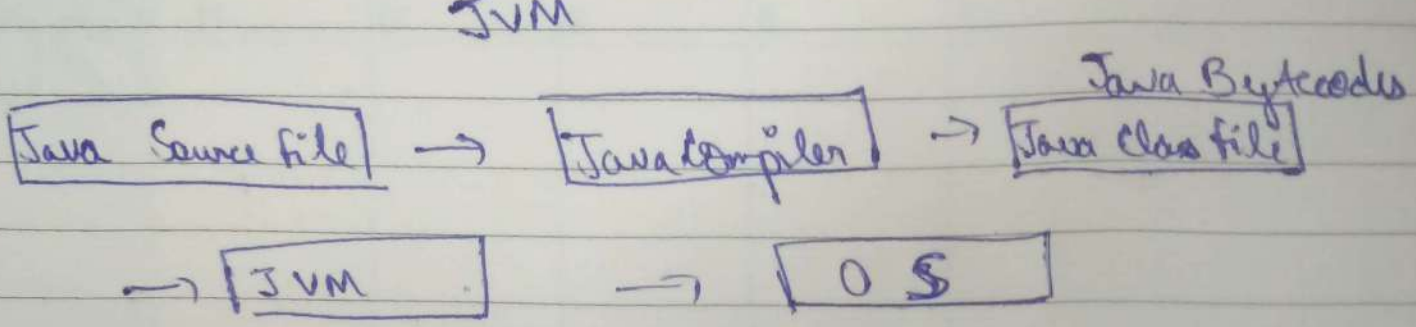
APRIL

APRIL - 2020						
M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

println vs print
=> println will insert a new line while print will simply print in the same line.

JVM (Java Virtual Machine)

=> It's responsible for converting bytecode to machine-specific code and is necessary in both JDK & JRE.



Pg 43

```
class DrumKit {
    boolean topHat = true;
    boolean snare = true;

    void playTopHat() {
        System.out.println("ding ding da-ding");
    }
    void playSnare() {
        System.out.println("bang bang ba-bang");
    }
}

class DrumKitTestDrive {
    public static void main(String[] args) {
        DrumKit d = new DrumKit();
        d.playSnare();
```

```
if (d.snore == true ) {
        d.playSnore();
}
}
}
```

Pg = 45

(1) I am compiled from a .java file - class

(2) My instance variable values can be different from my buddy's values - object

(3) I behave like a template - class

(4) I can have many methods - class, object

(5) I represent 'state' - instance variable

(6) I have behaviours - object, class

(7) I live on the heap - object

(8) I can change at runtime - object instance variable

2020

03 WK 14 (094-272)
FRIDAY
APRIL

APRIL - 2020						
M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

4.3

Primitive Types: boolean true / false

char 16 bits

integer 8 bits

short 16 bits

long 64 bits

floating point

float 32 bits

double 64 bits

3 steps of object, declaration, creation & assignment

① ② ③

→ Dog myDog = new Dog();

→ Create an Object: Dog myDog = new Dog();

→ Link the object & the reference :

Declare an array

① int[] nums;

② nums = new int[7];

③ nums[0] = 1; nums[2] = 44;

nums[1] = 19; nums[3] = 42;

Page 5 42 84 (class 'xcopy' compiles & runs as it stands.

Pg- 89 A class can have any number of these - interface, getter, setter

A method can have only one of these - return

This can be implicitly promoted - return, argument

It really means 'make a copy' • pas by value

I can have many arguments - method

I always fly solo - return

Pg-99 Prep Code → Test code → Real code

Pg-111 `int randomNum = (int) (Math.random() * 5)` (0-4.99

↑ We declare an int variable to hold the random number we get back

↑ A class that comes with JAVA

`String guess = helper.getUserInput("enter a number");`

↓ an instance

Pg 116 The enhanced for loop

`for(String name : nameArray) { }`

The colon (:) means "IN"

↑ The element in the array must be compatible with the declared variable type

05 WK 14 (096-270)
SUNDAY
APRIL

		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Pg 111

Converting a String to an int

```
int guess = Integer.parseInt(stringGuess);
```

Pg 119

```
class Multifor {
    public static void main(String[] args) {
        for (int x=0; x<4; x++) {

        for(int x=0; x<4; x++){
            for(int y=4; y>2; y--) {
                System.out.println(x + " " + y);
            }
        if (x==1) {
            x++;
        }
    }
}
```

Output

```
0 4
0 3
1 4
1 3
3 4
3 3
```

Pg 133 ArrayList

add(Object elem)
Add the object parameter to the list

remove(int index)
Removes the object at the index parameter

remove(Object elem)
remove this object

contains(Object elem)
Returns 'true' if these match

isEmpty()
Returns 'true' if this is not

```
import java.util.ArrayList;
public class DotCom {
    private ArrayList<String> locationCells;
    public void setLocationCells(ArrayList<String> loc) {
        locationCells = loc;
    }
public String checkYourself(String userInput) {
    String result = "miss";

    int index = locationCells.indexOf(userInput);
    if (index >= 0) {
            locationCells.remove(index);
    if (locationCells.isEmpty()) {
        result = "kill";
    } else {
        result = "hit";
    }
    }
    result result;
}
}
```

2020

07

WK 15 (098-268)

TUESDAY

APRIL

APRIL - 2020

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

JAVA API

Pg-161

```
import java.util.*;
public class ArrayListMagnet {
    public static void main (String[] arg) {
        ArrayList<String> a = new ArrayList<String>();
        a.add(0, "zero");
        a.add(1, "one");
        a.add(2, "two");
        a.add(3, "three");
        printAl(a);
        if (a.contains("three")) {
            a.add("four");
        }
        a.remove(2);
        printAl(a);
    }
    public static void printAl(ArrayList<String> al) {
        for (String element : al) {
            System.out.print(element + " ");
        }
        System.out.println(" ");
    }
}
```

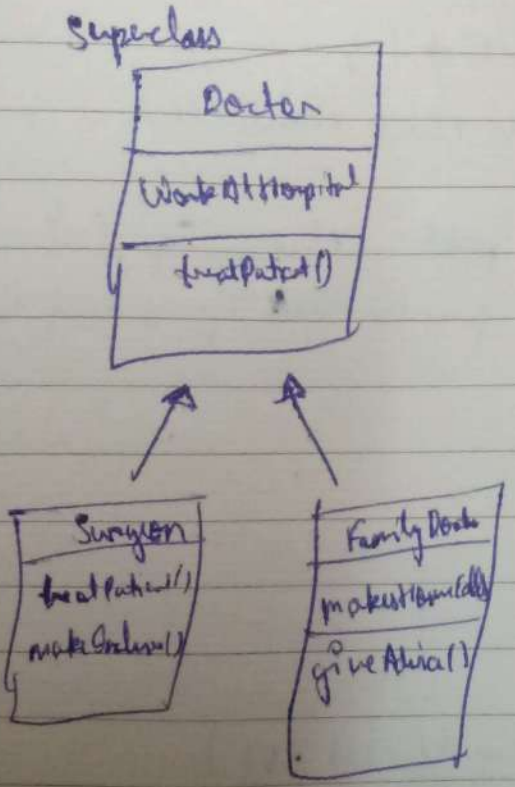
APRIL

Inheritance

→ When one class inherits from another, the subclass inherits from the superclass
→ the subclass extends the superclass

The four access levels: private default protected public

public members are inherited
private members are not inherited.



Polymorphism

When you are designing your class inheritance structure, you have to decide which classes are abstract & which are concrete.

```
abstract class Canine extends Animal {
    public void roam() { }
}
```

09 WK 15 (100-266)
THURSDAY
APRIL

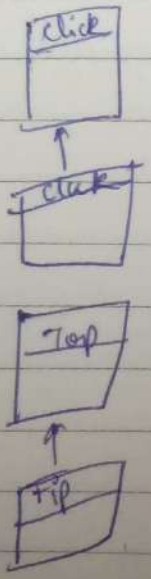
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

abstract class means that nobody can ever make a new instance of that class.

```
abstract public class Animal extends Animal
{
    public void roam(){ }
}
```

Interface & Polymorphism

Pg 251




```
public class Click { }
public class Clock extends Click { }

public abstract class Top { }
public class Tip extends Top
```

Pg 286

Math.round ()

```
int x = Math.round(-24.8f);
int y = Math.round(24.4f);
```

↑ Numeric, floating point literals are assumed to be double unless you add the 'f'

Math.min ()

```
int x = Math.min(24, 2401);
double y = Math.min(9087, 90826.491);
```

Formatting

Pg 194

```
public class TextFormats {
    public static void main (String[] args) {
        String s = String.format("%d", 1000000000);
        System.out.println(s);
    }
}
```

Output : 1,000,000,000

%d -> insert commas & format the number as a decimal integer

%.2f -> format the no. as a floating point with a precision of two decimal places.

%,.2f -> insert commas & format the no. as floating point with a precision of two decimal places

Pg 351

```
class MyEx extends Exception { }
public class ExTraDrive {
    public static void main (String[] args) {
        String test = args[0];
        try {
            System.out.print("t ");
            doRisky(test);
            throw new MyEx();
        }
        System.out.println("r ");
    }
}
```