Hello how are you

My SQL

USE

SELECT (*) show all column

FROM

WHERE

ORDER BY

-- (comment)

DISTINCT

AS to change name " "

Exercise — name products — unit price — new price (unit price * 1.1)

---

* Java was developed by James Gosling in 1995
* Orginally - Oak → Green → Java
* Java Edition
  - Standard Edition (SE)
  - Enterprise Edition (EE)
  - Micro Edition (ME)
  - Java Card

Fundamentals →
- Types
- Control
- Clean Code
- Finding & Fixing Errors
- Packaging your applications

Java

Function Block of code that performs a task

parameter of fun
↑
Return type Name() {.....}
↓ ↓ ↓
return type of fun name of fun code

* In java we put left brace ({) in same line we define fun.
* In java program should have atleast 1 main() function

Class A container for related function

class Main { }
name of class

Method fun which is part of class

Access modifier Keyword which tells or determine ~~that~~ if other classes or methods in this program can access these methods or classes.

```
public class Main {
    private void main() {
        ....
    }
}
```


Comment // Write the code //

Naming Classes : Pascal Naming Convention
Methods : Camel Naming Convention

First Java program

```
1. package com.Anuradha;
2.
3. public class Main {
4.
5.     public static void main(String[] args) {
6.         System.out.println("Hello World");
7.     }
8. }
```

Output: Hello World

Run Compilation + Execution

Source Code
↓
Java Compiler
↓
Byte Code → JVM → Native Code

Temporarily store the data

Variables:

```
1  package com.Base pac Anuradha;
2
3  public class main {
4      public static void main (String[]args{
5          int age = 30;
6          System.out.println("Hello" + age);
7      }
8  }
```

Output: 30

Ex. (i) int age = 30, temperature = 45;

(ii) int age = 30;
int heritage = age; (Camel Case Notation)

Primitive Types → for storing simple values

Reference Types → for storing complex objects

Primitive Types

Type	Bytes	Range
(i) byte	1	[-128, 127]
(ii) short	2	[-32K, 32K]
(iii) int	4	[-2B, 2B]
(iv) long	8	[ ]
(v) float	4	[ ]
(vi) double	8	[ ]
(vii) char	2	A, B, C, ..
(viii) boolean	1	True / False

Single letter ' '
String of letter " "


```
1  package com.codewith Anuradha;
2  public class Main {
3      public static void main(String[]args){
4          byte age = 30;
5          int viewsCount = 123456789;
6          long followers = 123-456-789-01L;
7          float price = 10.99F;
8          Char letter = 'A';
9          boolean isEnglish = false;
10     }
11 }
```

Can't written in "/"

Reference Type

*We have to allocate the memory.
(We don't have to release these memorys JRE will automatically take care of that)

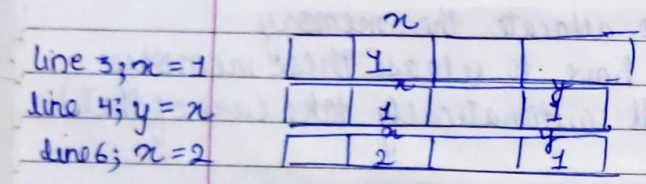
```
1  package com.Anuradha;
2  public import java.util.Date;
3  public class Main {
4      public static void main(String[] args){ {
5          Date now = new Date();
6          System.out.println(now);
7      }
8  }
9
```

Output: Current D-M-Y.
Day Month Date Time T.Z. Year
(Time Zone)

Primitive VS Reference

```
1  package com.Anuradha;
2  public class Main {
3      public static void main(String[] args){
4          byte x = 1;
5          byte y = x;
6          x = 2;
7          System.out.println(y);
8          System.out.println(x);
9      }
10 }
```

Output : 1
2

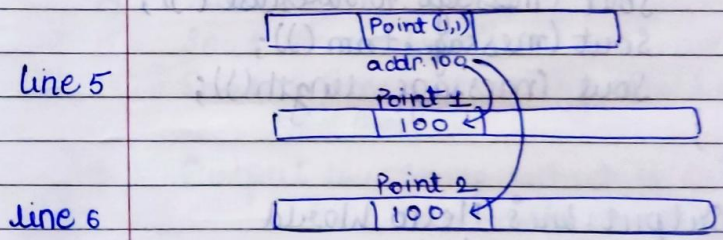


completly independent of each other

Reference : Don't store the actual value but reference of that object


```
1.  package com.Anuradha;
2.  import java.awt.*;
3.  public class Main {
4.      public static void main(String[] agrs) {
5.          Point point1 = new Point(1,1);
6.          Point point2 = point1;
7.          point1.x = 2;
8.          sout(point1);
9.          sout(point2);
10.     }
11. }
```

Output : java.awt.Point[x=2,y=1]
java.awt.Point[x=2,y=1]



not independent

String

```
1  package cos.Anuradha;
2  public class Main{
3      public static void main(String[]agrs){
4          String message = "HelloWorld"+"!!";
5
6  // short for: new String("HelloWorld"+"!!");
7
8          Sout(message);
9          Sout(message.endsWith("!!"));
10         Sout(message.startsWith("!!"));
11         Sout(message.stIndexof("H"));
12         Sout(message.indexof("45"));
13         Sout(message.replace("!","5");
14
15 // not changes the orignal string //
16
17         Sout(message.toLowerCase());
18         Sout(message.trim());
19         Sout(message.length());
20     }
21 }
```

Output: line 8. Hello World
line 9. True
line 10. false
line 11. 0
line 12. -1 (for those which not exsist)
line 13. HelloWorld 5 5
line 17. hello world!!
line 18. Hello World!!
line 19. 13

new line \n
tab \n

Escape sequence

```
Escape  Escape  String message = "Hello \"Anu\"";
                Sout(message);
```

Output Hello" Anu"

```
String message = "C:\\window\...";
Sout(message);
```

Output C:\window\...

Arrays

Store list of no., items, people etc.

```
4  import java.util.Arrays;
9  int[] numbers = new int[5];
10 numbers[0] = 1;
11 numbers[1] = 2;
12 numbers[2] = 3;          //Array.sort(numbers)//
13 numbers[3] = 4;
14 numbers[4] = 5;
15 Sout(numbers);
16 Sout(Arrays.toString(numbers));
       to print Array
```

Output line 15: String which is calculated base on the address of this object is memory
line 16: [1, 2, 3, 4, 5]

```
9  int[] num = {1, 2, 8, 3, 4}
10 Sout(Arrays.toString(num));
11 Sout(num.length);
```

Output: 10. [1, 2, 8, 3, 4]
11. 5

Multidimensional Arrays

row column

9. int [][] numbers = new int [2][3]
10. numbers[0][0] = 1;
11. numbers [1][2] = 6;
12. Sout (Arrays.deepToString(numbers));

Output [[1,0,0],[0,0,6]]

9. int [][] numbers = {{1,2,3},{4,5,6}};
10. Sout (Arrays.deepToString(numbers));

Output [[1,2,3],[4,5,6]]

9. int [][][] numbers = new int [3][2][2]
10. numbers [0][0][0] = 1;
11. numbers [0][0][1] = 2;
12. numbers [0][1][0] = 3;
13. numbers [2][1][1] = 12;
14. Sout (Arrays.deepToString (numbers));

Output [[[1,0],[3,0]],
[[0,0],[0,0]],
[[0,0],[0,12]]]

9. int [][][] numbers = {{{1,2},{3,4}},{{5,6},{7,8}},
{{9,10},{11,12}}};
Sout (Arrays.deepToString(numbers));

Output [[[1,2],[3,4]],[[5,6],[7,8]],
[[9,10],[11,12]]]

Constants

6. final float pi = 3.14F
7. pi = 1;

then Java will show error

Airthmatic Expressions

6. int result = 10 + 48;
7. Sout (result);
Output: 58

6. int result = 10/3;
7. Sout (result); // division of 2 whole no. is // whole no. //
Output: 3

6. double result = (double) 10 / (double) 3;
7. Sout (result);
Output 3.3333333333333335

order of operation
()
*/
+-

Casting

Implicit casting no data loss
byte > short > int > long > float > double

double x = 1.1;
int y = (int) x + 2;
Sout (y);
Output : 3

→ Explicit casting
Data loose

ceiling : smallest integer which is equal or greater
floor : largest integer which is equal ~~or~~ smaller

Page No.

```
6. // Explicit Casting
7. String x = "1.1";
8. float y = Float.parseFloat(x) + 2;
9. Sout (y);
```
Output : 3.1

The Math Class

```
6. int x = Math.round(1.1F);
7. Sout (x);
8. int result = (int) Math.ceil(1.1F);
9. Sout (result);
10. int y = (int) Math.floor(1.1F);
11. Sout(y);
```
Output line 7. 1
9 2
11. 1

Math.round(); Math.min/max(); Math.random()

Note: Abstract class can't be instantiated

Formatting Numbers

Why new is not used

```
3 import java.text.NumberFormat;
8. NumberFormat currency = NumberFormat.
                    getCurrencyInstance();
9. String result = currency.format(123456.89);
10. Sout(result);
```
Output : $ 123,456.89

Shift+F6 = Refactor

Date

Page No. 7

```
8. NumberFormat percent = NumberFormat.
                    getPercentInstance();
9. String result = percent.format(0.70);
10. Sout (result);
```
Output: 70%

Or

```
8. String result = NumberFomat.getPerantInstance().
                    format(0.70);
Sout (result);
```
Output : 70%.

Read Input

```
3. import java.util.scanner;
4. Public class void {
    public static void main (String[] args) {
      Scanner scanner = new Scanner(System.in);
      byte age = scanner.nextByte();
      Sout ("You are " +age);
```

Comparison Operators

```
1 package com.Anuradha;
2 public class Main {
3   public static void main (String [] aggrs) {
4     int x = 1;
5     int y = 1;
6     Sout (x == y);
7   }
8 }
```

Result / Output: true

== > != < >= <=

Logical Operators

```
4. int temperature = 22;
5. boolean isWarm = temperature > 20 && temperature < 30;
6. Sout (isWarm);
```

Output: true

&& || !

If statements

```
1  package com. Anuradha
2  public class Main {
3    public static void main (String[] args){
4      int temp = 32;
5      if ( temp > 30) {
6        Sout (" It's a hot day");
7        Sout (" Drink Water");
8      } else if (temp > 20 && temp <=30)
9        Sout (" Beautiful day");
10     else
11       Sout (" Cold day");
12   }
13 }
```

Output: It's a hot day
Drink Water


```
1  package com. Anuradha;
2  public class Main {
3    public static void main (String [] args){
4      int income = 120-000;
5      boolean hasHighIncome;
6
7      if (income > 100-000)
8        hasHighIncome = true;
9      else
10       hasHighIncome = false;
11   }
12 }
```

↓ (same as above code)

```
4.     int income = 120-000;
5.     boolean hasHighIncome = (income > 1000);
6.   }
7. }
```

Ternary Operators

Expression ? If true, print : If false, print;

```
4.     int income = 120-000;
5.     String className = income > 1000 ? "Fir" : "Sec";
6.   }
7. }
```

Switch statement

```
1. package com.Anuradha
2. public class Main {
3.     public static void main(String[]args){
4.         String role = "admin";
5.         if (role == "admin").
6.             Sout ("you are admin");
7.         else if (role == "moderator")
8.             Sout ("you are moderator);
9.         else
10.            Sout ("you are guest);
11.    }
12. }
```

```
5.  switch(role) {
6.      case "admin":
7.          Sout ("you are admin");
8.          break;
9.      case "moderator":
10.         Sout ("you are moderator");
11.         break;
12.     default:
13.         Sout("you are guest");
14. }
```

for loop

```
for (initialization; condition; updation)
```

```
5. for(int i=0; i<5 ; i++)
6.     Sout ("Hello World");
```


While loop

```
while (condition) {
    // body
    updation;
}
```

```
5. int i = 0;
6. while(i<5) {
7.     Sout("Hello World");
8. }i++;
9. }
```

Example: To take input until user type quit

```
1. package com.Anuradha;
2. import java.util.Scanner;
3. public class Main {
4.
5.   public static void main(String[] args) {
6.     Scanner scanner = new Scanner(System.in);
7.     String input = " ";
8.
9.     while (!input.equals("quit")) {
10.      sout ("Input: ");
11.      input = scanner.nextLine().toLowerCase();
12.      Sout (input);
13.    }
14. }
```

Do While

```
1  package com.Anuradha;
2  import java.util.Scanner;
3  public class Main {
4      public static void main(String[] args){
5          Scanner scanner = new Scanner(System.in
6          String input = " ";
7
8          do {
9              Sout(" Input: ");
10             Input = Scanner.nextLine().toLowerCase
11             Sout(input);
12         }
13         while (!input.equals("quit"));
14     }
15 }
```

```
do {
    // body
}updation;}
while (condition);
```

continue — Send back to the starting of the loop

```
continue;
```

break — Send out of the loop

```
break;
```


for each loop

```
5.  for String[] fruits = {"Apple", "Mango", "Orange"};
6.
7.  for(int i= 0; i < fruit.length; i++)
8.      Sout (fruits[i]);
```

```
5.  String[] fruits = {"Apple", "Mango", "Orange"};
6.
7.  for (String fruit : fruits)
8.      Sout (fruit);
```

* can't access index
* can't access back to front

Project 1: Mortgage calculator

```
1. package com.Anuradha
2. public class Main {
3.     public static void main(String[] args) {
```

Date
Page No.

Project 1 — Mortgage Calculator

```
1. package com.Anuradha;
2. import java.text.NumberFormat;
3. import java.util.Scanner;
4. public class Main{
5.     public static void main(String[] args) {
6.         final byte MONTHS_IN_YEAR = 12;
7.         final byte PERCENT = 100;
8.
9.         Scanner scanner = new Scanner(System.);
10.
11.        Sout("Principle : ");
12.        int principle = scanner.nextInt();
13.
14.        Sout("Annual Interest Rate");
15.        float aInnual Interest = s$canner.nextFloa;
16.        float monthlyInterest = annulIntrest
17.                                / Percent
18.                                / MONTHS_IN_yEAR;
19.        Sout(" Period (years):");
20.        byte years = scanner.nextByte();
21.        int noOfPayments = years *
                                MONTHS_IN_YEAR;
22.        double mortgage = principle
                * (monthlyInterest * Math.Pow(1+ monthlyIntrest, noOfPayments))
                / (Math.pow(1 + monthlyIntrest, noOfPayments) -1);
23.        String mortgageFormatted = NumberFormatted.getCurrencyInstance().format(mortgage)
24.        Sout("Mortgage : " + Mo mortgageFormatted);
25.    }
26.}
```

Date
Page No. 1

$$\text{Mortgage} = P \frac{r(1+r)^n}{(1+r)^n - 1}$$

P = principle
r = Rate monthly
n = time / no. of payments

Project 2 Mortgage Calculater with Range

```
1. package com.Anuradha;
2. import java.text.NumberFormat;
3. import java.util.Scanner;
4. public class Main {
5.     public static void main(String[] args){
6.         final byte MONTHS_IN_YEAR = 12;
7.         final byte PERCENT = 100;
8.         int principle = 0;
9.         float monthlyInterest = 0F;
10.        int numberOfPayment = 0;
11.
12.        Scanner scanner = new Scanner(System.in);
13.
14.        while (true){
15.            sout("Principle : ");
16.            principle = scanner.nextInt();
17.            if(principle >=1000 && principle <= 1_000_000)
18.                break;
19.            else
20.                sout("Enter a no. b/w 1,000 and 1,000,000");
21.        }
22.
23.        while(true){
24.            sout("Annul Interest Rate");
25.            float annulInterestRate = scanner.nextFloat();
26.            if(annulInterestRate >=1 && annulInterestRate <= 30){
27.                monthlyInterest = annulInterestRate / PERCENT / MONTHS_IN_YEAR;
28.                break;
29.            }
```

```
30.  $      Sout ("Enter value between 1 to3
31.         }
32.
33.      While (true) {
34.          Sout ("Period (years) : ");
35.          byte years = scanner.next Byte
36.          if (years >1 && years <=30) {
37.              numberOf Payment = years *
38.              break;
39.          }
40.          Sout ("Enter years between 1 to30");
41.      }
42.
45.      double mortgage = principle
44.                  * (monthly Intrerest *
45.                  / (Math.pow (1+ monthly
46.
47.      String  mortgageFormatted = Nu
48.      Sout ("Mortgage :" + mortgageFo
49.      }
50. }
```


"Any fool can write code that computer can understand. Good Programmer write code that humans can understand."

-Martin Fowler

```
MONTHS_IN_YEARS;

;

Math.pow (1+ monthly Interest, numberOf Payment))
nterest, numberOf Payment) -1);

berFormat.get Currency Instance ().format (mortgage);
rmatted);
```

Creating Method

```
1  package com.Anuradha;
2  public class Main {
3      public static void main(String[] args) {
4          greetUser("Mosh");
5          greetUser("John");
6      }
7      public static void greetUser(String name) {
8          System.out.println("Hello " + name);
9      }
10 }
```

Output :
Hello Mosh
Hello John

Refactoring

Changing the structure of code without changing its behavior

```
1  package com.Anuradha;
2  public class Main {
3  import java.util.Scanner;
4  import java.text.NumberFormat;
5  public class Main {
6      public static void main(String[] args) {
7          int principle = 0;
8          float annualInterest = 0;
9          byte years = 0;
10
11         principle = (int) readNumber("Principle", 1000, 1000000);
12         annualInterest = (float) readNumber("Annual Interest", 1, 30);
13         years = (byte) readNumber("Years", 1, 30);
14         double mortgage = calculateMortgage(principle, annualInterest, years);
15
16         String mortgageFormatted = NumberFormat.getCurrencyInstance().format(mortgage);
17         System.out.println("Mortgage : " + mortgageFormatted);
18     }
19     public static double readNumber(String prompt,
20                                     double min, double max) {
21         Scanner scanner = new Scanner(System.in);
22         double value;
23         while (true) {
24             System.out.println(prompt + " : ");
25             value = scanner.nextFloat();
26             if (value >= min && value <= max)
27                 break;
28             System.out.println("Enter value b/w " + min + " and " + max);
29         }
30         return value;
31     }
32     public static double calculateMortgage(int principle,
33                                            float annualInterest, byte years) {
34         final byte MONTHS_IN_YEARS = 12;
35         final byte PERCENT = 100;
36         float monthlyInterest = annualInterest / PERCENT / MONTHS_IN_YEARS;
37         short numberOfPayment = (short) (years * MONTHS_IN_YEARS);
38         double mortgage = principle
39             * (monthlyInterest * Math.pow(1 + monthlyInterest, numberOfPay))
40             / (Math.pow(1 + monthlyInterest, numberOfPayment) - 1);
41         return mortgage;
42     }
43 }
```

readNumber class

calculateMortgage class

Calling of readNumber

Mortgage Calculator

Project 5 Payment Schedule

```
1  package com.Anuradha;
2
3  import java.util.Scanner;
4  import java.text.NumberFormat;
5
6  public class Main {
7      final static byte MONTHS_IN_YEAR = 12;
8      public static void PERCENT = 100;
9
10     public static void main(String[] args){
11         int principle = 0;
12         float annualInterest = 0;
13         byte years = 0;
14
15         principle = (int)readNumber("Principle", 1000, 1,000,000);
16         annualInterest = (float)readNumber("Annual Interest Rate", 1, 30);
17         years = (byte)readNumber("Period (years)", 1, 30);
18
19         PrintMortgage(principle, annualInterest, years);
20         PrintPaymentSchedule(principle, annualInterest, years);
21     }
22
23     private static void printMortgage(int principle, float annualInterest, byte years){
24         double mortgage = calculateMortgage(principle, annualInterest, years);
25         String mortgageFormatted = NumberFormat.getCurrencyInstance().format(mortgage);
26         Sout();
27         Sout("MORTAGE");
28         Sout("--------");
29         Sout("Monthly Payments: " + mortgageFormatted);
30     }
```

print mortgage class

```
31. private static void printPaymentSchedule
32.     Sout();
33.     Sout("PAYMENT SCHEDULE");
34.     Sout("-----------------");
35.     for (int/short month = 1; month <= years
36.         double balance = calcuteBalance
37.         Sout String Bbalance formatted =
38.         Sout(balanceformatted);
39.     }
40. }
41.
42. public static void double readNumber(
43.     Scanner scanner = new Scanner(Syst
44.     double value;
45.     while(true){
46.         Sout(prompt + ": ");
47.         value = scanner.nextFloat();
48.         if (value >= min && value
49.             break;
50.         Sout("Enter value b/w " + min
51.     }
52.     return value;
53. }
54. public static double calculateBaBlance
55.     float monthlyInterest = annual
56.     float short number Of Payment = (short)
57.     double balance = principle
58.     return balance;
59. }
```

Print Payment Schedule class

read value class

calculate balance class

```
(int principle, float annualInterest, byte years){
* MONTHS_IN_YEAR; month++){
(principle, annualInterest, years, month);
NumberFormat.getCurrencyInstance().format(balan
String prompt, double min, double max){
n.in);
<=max)
+ " and " + max);
(int principle, float annualInterest, byte years,
    short no number Of Payment Made){
interest / PERCENT / MONTHS_IN_YEARS;
(years * MONTHS_IN_YEAR);
* Math.pow(1+monthlyInterest, number of Payment) - Mathpow(1+MonthlyInterest, number of PaymentMade)
/(Math.pow(1+monthlyInterest, number of Payment) - 1);
```

stack overflow
for Run time error

Date
Page No.

```
60. public static double calculateMortgage(
61.     float monthlyInterest = annualInte
62.     short numberOfPayment = (short) (
63.
64.     double mortgage = principle
65.             * (mothly Interest
66.             / (Math.pow(1 + mo
69.     return mortgage;
68. }
69.}
```

return mortgage class

int principle, float annualInterest, byte years) {
/PERCENT / MONTHS_IN_YEAR;
= * MONTHS_IN_YEAR);

Math.pow(1 + monthlyInterest, numberOfPayment))
hly Interest, numberOfPayment) - 1);

Error (i) Compile-time error } easy to cover
(ii) Syntax error ↑ happens } When we don't follow syntax
(iii) Run-time error (Debugger) Tool required

Date
Page No. 17

OOP

Date
Page No.

Programming Paradigms

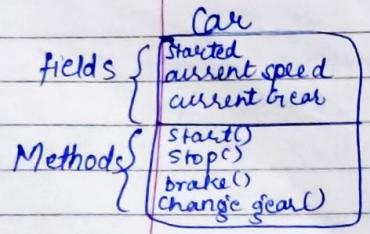
way of writing programming language

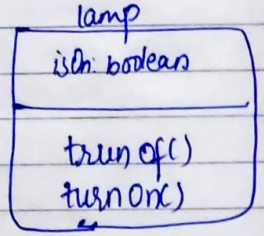
- Procedural
- Event-drive
- Functional
- Logic
- Object-oriented
- Aspect-oriented

Object-oriented

Benefits: Reduced complexity
Easier maintenance
Code reuse
Faster Development

Classes and Object





throw new IllegalArgumentException("can't take this value.")

Date
Page No. 18

Encapsulation Bundle the data and methods that operate on the data in a single unit

```
3. public class Main {
4.     public static void main(String[] args) {
5.         var employee = new Employee();
6.         employee.baseSalary = 50.000;
7.         employee.hourlyRate = 20;
8.         int wage = employee.calculateWage(10);
9.         Sout(wage);
10.    }
11. }
```

```
3. public class Employee {
4.     public int baseSalary;
5.     public int hourlyRate;
6.
7.     public int calculatedWage(int extraHours) {
8.         return baseSalary + (hourlyRate * extraH);
9.     }
10. }
```

- curs

getters & setters

```
3. public class Employee {
10.    public void setBaseSalary(int BaseSalary) {
11.        this.baseSalary = baseSalary;
12.    }
13.    public int getBaseSalary() {
14.        return baseSalary;
15.    }
16. }
```

Classes: A blueprint for creating objects.

Object: An instance of a class

UML : Unified modeling language

Creating new class

```
1  package com.code.withmosh;
2
3  public class TextBox {
4      public String text; // Field
5
6      public void set_text(String text) {
7          this.text = text;
8      }
9
10     public void clear() {
11         text = "";
12     }
13 }
```

```
1 package com.code.withmosh;
2 public class Main {
3     public static void (String[] args) {
4         var textbox1 = new TextBox();
5         textbox1.set_text("Box 1");
6         sout (text_box1.text);
7     }
8 }
```


memory allocation

```
4         var text_box1 = new TextBox();
5         var text_box2 = text_box1;
6         text_box2.set_text("Box 1");
7         sout (text_box1.text);
8     }
9 }
```

Output : Box 1

Getters & Setters

```
1. package com.codewithmosh;
2.
3. public class main Employee {
4.     private int base_salary;
5.     private int hourly_rate;
6.
7.     public int calculate_wage(int extra_hours){
8.         return base_salary + hourly_rate * extra_
                                              hours );
9.     }
10
11.    public void setBase_salary (int Base_salary){
12.        if (base_salary <= 0)
13.            throw new IllegalArgumentExcep
                -tion ("Salary can't be 0 or less");
14.        this.base_salary = base_salary; }
15
16.    public int getBase_salary() {
17.        return base_salary;
18.    }
19.    public void setHourly_rate(...
25.    public void getHourly_rate(...
27 }
28
```

Abstraction Reduce complexity by hiding unnecessary details.
(implementation details)

Coupling The level of dependency b/w classes

Constructor

```
1  package com.codewithmosh;
2
3  public class Employee {
4      private
5      private
6
7      public Employee (int base_salary, int hourly_rate
8          set Base_salary ( base_salary );
9          set Hourly_rate (hourly_rate);
10     }
⊖
⊖
⊖      private void set Base_salary ( )
⊖      private int get Base_salary ( )
⊖      private void set Hourly_rate ( )
⊖      private int getHourly_rate( ) ...
```

If we don't ~~construct~~ create constructor then java itself create a constructor with default value for each field.

default value ↓
boolean = false
int = 0.

Method overloading If a class have multiple methods having same name but different in parameter.

Constructor Overloading having more than 1 constructor with different parameter lists in a class

Static members:

Mortgage Calculator.

Main Class →

```
1. package com.codewithmosh;
2.   public class Main {
3.       public static void main (String[] args) {
4.       int int principle =0;
5.           float annut interest rate = 0;
6.           byte years =0;
7.
8.           principle = (int)Console.readNumber ("principle", 1000, 1000.000);
9.           annul..... =(float) ''      ("Annul Interest", 1, 30);
10.          years = (byte) '' ("years", 1, 30);
11.
12.          var calculator = new MortgageCalculator(p, ai, y)
13.
14.          var report = new MortgageReport( calculator);
15.          report.print_mortgage();
16.          report.print_payment_schedule();
```

Console Class →

```
1.
2. import java.util.Scanner;
3. public class Console {
4.     private static Scanner scanner = new Scanner(System.in);
5.
6.     public static double readNumber (String promt){
7.         return scanner.nextDouble();
8.     }
9.
10.    public static double readNumber (String prompt,
11.                                     double min,
12.                                     double max) {
13.        double value;
14.        while (true){
15.            Sout (promt + " :");
16.            value : scanner.nextDouble();
17.            if (value >= min && value <= max)
18.                break;
19.            Sout (" Enter value b/w "+ min +'&"+max);
20.        }
21.        return value;
22.    }
```

MortgageCalculator Class →

```
1.
2. public class MortgageCalculator {
3.     private final static byte MONTHS_IN_YEAR =12;
4.                            PERCENT =100;
5.
6.     private int principle;
7.     private float annual_interest_rate;
8.     private byte years;
9.
10.    public MortgageCalculator(int principle,
11.                              float annual_interest_rate,
12.                              byte years) {
13.        this.principle = principle;
14.        this.annual_interest_rate = annual_interest_rate;
15.        this.years = years;
```

```
16.  public double calculate_mortgage(){
17.      float monthly_interest_rate = getMonthly
18.      short number_of_payments = getNumber
19.
20.      double mortgage = principle
21.                  *(monthly_interest_rate
22.                  /(Math.pow(1+monthly
23.      return mortgage;
24.  }
25.  public double calculate_balance(short nu
26.      float monthly_interest_rate = getMonth
27.      short numbers_of_payments = getNumber
28.
29.      double balance = principle
30.                  *(Math.pow(1+monthly
31.                  number_of_payments_ma
32.      return balance;
33.  }
34.
35.  public double[] get_remaining_balances(){
36.      var balances = new double[getNumb
37.      for(short month=1; month<=balance
38.          balances[month-1] = calculateB
39.      }
40.      return balances;
41.  }
42.  private float getMonthly_interest_rate(){ret
43.  private short getNumber_of_payments(){
49.  }
```

```
rest_rate();
payments();

th.pow(1+monthly_interest_rate, number_of_payments))
(rest_rate, number_of_payments)-1);

er_of_payments_made){
interest_rate();
payments();

rest_rate, number_of_payments) - Math.pow(1+monthly_inter
est_rate,
/(Math.pow(1+monthly_interest_rate, number_of_payments)-1));

of_payments()];
ngth; month++){
nce(month);

annual_interest_rate / PERCENT / MONTHS_IN_YEARS;}
return (short)(years * MONTHS_IN_YEARS);}
```

MortgageReport class =>

```
1. package com.codewithmosh;
2. import java.text.NumberFormat;
3. public class MortgageReport {
4.     private final NumberFormat currency;
5.     private MortgageCalculator calculator;
6.
7.     public MortgageReport(MortgageCalculator
8.         this.calculator = calculator;
9.         currency = NumberFormat.getCurrencyIns
10.    }
11.
12.    public void printMortgage() {
13.        double mortgage = calculator.calculate
14.        String mortgageFormatted = currency.for
15.        Sout();
16.        Sout("MORTAGAGE");
17.        Sout("----------");
18.        Sout("Monthly Payments:" + mortgageFo
19.    }
20.
21.    public void printPaymentSchedule() {
22.        Sout();
23.        Sout("Payment Schedule");
24.        Sout("----------");
25.        for(double balance : calculator.getRun
26.            Sout(currency.format(balance));
27.    }
28. }
```

Java not support multiple inheritance.

Inheritance When one class acquires the properties of another class

Upcasting Casting an object to one of its super type

Downcasting " " " " " " sub types.

Hashcode It is an integer value that is associated with each object in java. It facilitate hashing in hash tables

Method overriding `@Override` must have a same signature. It allow subclass to provide specific implementation of method that is already provided by parent class

Poly-morph-ism Many form
It occurs when we have many classes that are related to each other by inheritance. Allow us to perform single action in ~~many~~ form different ways.

Abstract class
- Must contain one abstract method
- Can't instantiate it only extends it
- Parent class has a named method, but need its child class to fill out the tasks (To share code)

Interfaces To build loosely-coupled, extensible, testable application