Le programme AFPA

La formation se compose de 6 modules, complétés par 1 période en entreprise.

Période d'intégration. Accueil, présentation des objectifs de formation, connaissance de l'environnement professionnel, sensibilisation au développement durable, adaptation du parcours de formation **(1 semaine).**

Module 1. Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité: Développement d'une interface utilisateur de type Desktop en pratiquant une veille éventuellement à partir de documentation en langue anglaise - maquettage d'une application - développement d'une interface utilisateur et des composants d'accès aux données **(9 semaines).**

Module 2. Développer une application Web en intégrant les recommandations de sécurité : développement de la partie front-end d'une interface utilisateur Web en effeectuant une veille technologique y compris en anglais.**(5 semaines).**

Module 3. Concevoir et developper la persistance des données en intégrant les recommandations de sécurité : conception et mise en place d'une base de données - développement des composants dans le langage d'une base de données **(4 semaines)**.

Module 4. Développer une application en couches en intégrant les recommandations de sécurité : participation à la gestion d'un projet informatique et à l'organisation de l'environnement de developpement - conception d'une application **(4 semaines).**

Module 5. Développer une application en couches en intégrant les recommandations de sécurité : developpement des composants métier - construction d'une applicationorganisé en couches , préparation et executiondes plans de tests d'une application - préparation et executionj du deploiement d'une execution **(8 semaines).**

Module 6. Développer une applicationmobile en intégrant les recommandations de sécurité : developpement d'une application en effectuant une veille technologique y compris en anglais **(2 semaines).**

Calendrier

CONCEPTEUR DEVELOPPEUR D'APPLICATIONS

Afpa

Du 28/09/2020 **au** 18/06/2021

Centre AFPA	:	8 rue Georges et Maï Politzer - 75012 Paris
Votre contact AFPA	:	elisabeth.cedovim@afpa.fr
Offre N°	:	20055
Durée totale du parcours	:	1261 Durée parcours
En formation à l'AFPA	:	968 heures (dont 31 heures pour la certification)
En entreprise	:	293 heures
Horaire des repas	:	de 12h30 à 13h30 *(pas de restauration sur le centre AFPA Paris)*
Horaire de formation *(sous réserve de modifications)*	:	du lundi au jeudi de 8h30 à 12h30 et de 13h30 à 17h15 le vendredi de 8h30 à 12h30

2020

Afpa - 2

SEPTEMBRE 23h

	L	M	M	J	V	S	D
36		1	2	3	4	5	6
37	7	8	9	10	11	12	13
38	14	15	16	17	18	19	20
39	21	22	23	24	25	26	27
40	28	29	30				

OCTOBRE 152h

	L	M	M	J	V	S	D
40				1	2	3	4
41	5	6	7	8	9	10	11
42	12	13	14	15	16	17	18
43	19	20	21	22	23	24	25
44	26	27	28	29	30	31	

NOVEMBRE 139h

	L	M	M	J	V	S	D
44							
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30						

DECEMBRE 129h

	L	M	M	J	V	S	D
49		1	2	3	4	5	6
50	7	8	9	10	11	12	13
51	14	15	16	17	18	19	20
52	21	22	23	24		26	27
53							

2021

JANVIER 140h

	L	M	M	J	V	S	D
53						2	3
1	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17
3	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31

FÉVRIER 140h

	L	M	M	J	V	S	D
5	1	2	3	4	5	6	7
6	8	9	10	11	12	13	14
7	15	16	17	18	19	20	21
8	22	23	24	25	26	27	28

MARS 163h

	L	M	M	J	V	S	D
9	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14
11	15	16	17	18	19	20	21
12	22	23	24	25	26	27	28
13	29	30	31				

AVRIL 12h + 133h PE

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

stage 2

MAI 132h PE

	L	M	M	J	V	S	D
17						1	2
18	3	4	5	6	7	8	9
19	10	11	12	13	14	15	16
20	17	18	19	20	21	22	23
21	24	25	26	27	28	29	30
22	31						

JUIN 28h PE + 70h

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

début et fin de la formation
présence en centre de formation
présence en entreprise (PE)
certification
congés, jours fériés

Paris, le 28/09/2020

Cette action est cofinancée par l'Union européenne. L'Europe s'engage en France avec le Fonds social européen.

28.09.20

Marc Lambert marc.lambert@afpa.fr

$8^{45} - 17^{15}$ } 35 heures
Vendredi: until 12:30

שש זה כנראה אנגלית

416 Alicia NACEUR $10^{00} - 12^{00}$ כיתה 315

407 Carole Djefri ליפול בגוטים שהבאו ביום הראשון. בלונדינית

Solidarite trasport תחבורה : הלוחה

Venneca Sultan עזרה עם הורדה או כל עזרה אחרת שקשורה ל- formation

Johnatan: Base de donner

הבחור עם המשקפיים שואל ליד ה~~בחור~~ זאת רוסייה מורה היא יודעת Java, C וכול מיני. כמה לגה מאוד

כלי: ← Scrum

שני	שלישי	רביעי	חמישי	שישי
הגשה	לקציע	לקציע	בסיס	קוסים

Metier Informatique

il faut chercher:

le mot technique, les diffrents metier, comportement(competence)

* SSI - securité de system d'info., "chartes d'usage d'Internet";
  DCSS - Direction centrale de la securité des sys d'info.
* architecture agile, spam, VPN, Algorithem, serveur, Hardware, reseaux, scrum
  Hacker,

ID: CDA213

reaseau → \\ares\USERS\CDA2019.315 תיקיית הקורס

APEC, monster.fr אתרים של חיפוש עבודה

Famille de informatique

- System
- Gestion ← אני — בית ספר מחשבים 42. ישאירו לך להסתדר לבד: epitech
- Jeu video
- Jeux : école : Goblin

IMS = ? Protocol de email

Competence

- travail en équipe
- Autonomie (1°)
- Curioisité
- persévérance & tenacité
- Imagination

Dynamisme → [illegible]
Rigeur
polyvalence

01.10.20

Algo

```
DEBUT

ENTIER I

// 1 octet : 2^8 256 0->255
// 1 octet sign : -128 -> +127
// 2 octet : 2^16 65536 0 -> 65535
// 2 octet : -32768 -> +32767

I <- 0  // I = 0

AFFICHER I      // ECRIRE I            -> 0
SAISIR I        // LIRE I

ENTIER J, K

J <- 2
K <- 3

I <- J + K      // 5
I <- J - K      // -1
I <- J * K  // 6
I <- J / K      // 0

SI I > J ALORS
        AFFICHER I
FINSI

SI I > J ALORS
        AFFICHER I
SINON
        AFFICHER J
FINSI

>
>=
<
<=
==      // =
!=      // <>   ≠

I <- 0 ->                   // INIT
TANTQUE I < 5               // TEST
        I <- I + 1          // INC
        AFFICHER I          // CORPS
FINTANTQUE

I
0
1       1
2       2
3       3
4       4
5       5
```

bit

byte

1 byte

2 bytes

if

2

```
I <- 0                          // INIT
FAIRE
        I <- I + 1              // INC
        AFFICHER I              // CORPS
TANTQUE I < 5           // TEST

POUR I ALLANT DE 0 JUSQUA 5 [PAS 1] // INIT TEST INC
        AFFICHER I                                              // CORPS
FINPOUR


FIN
```

02.10.20

Tableaux

```
DEBUT

TABLEAU ENTIER T[ 10]   // 1 à 10 ou 0 à 9 ou 1990 à 2020

ENTIER N
LIRE N
TABLEAU ENTIER TT[ N]

T[ 0] <- 10
LIRE T[ 0]
AFFICHER T[ 0]

AFFICHER T[ -1] // Tilt
AFFICHER T[ 11] // Tilt
AFFICHER T[ 10] // Tilt

AFFICHER LONGUEUR( T)          // 10

// LIRE( T)
ENTIER I
POUR I ALLANT DE 0 A LONGUEUR( T)-1
        LIRE T[ I]
FINPOUR

// AFFICHER( T)
POUR I ALLANT DE 0 A LONGUEUR( T) - 1
        AFFICHER T[ I]
FINPOUR

FIN
```

Loop : For

- Dichotomique :
- Séquentiel : לולאה.

Strings

```
CHAINE S
S <- "Bonjour"
AFFICHER S                // Bonjour
AFFICHER LONGUEUR( S)     // 7

CHAINE T <- "Comment ça va ?"

CHAINE V <- S & " " & T   //
AFFICHER V                // Bonjour Comment ça va ?
AFFICHER LONGUEUR( V)     // 24

ENTIER I
S <- "HELLO"
POUR I ALLANT DE 0 JUSQUA LONGUEUR( S) -1
        AFFICHER S[ I]
FINPOUR

H
E
L
L
O

AFFICHER S[5]   // Tilt

T <- ""
AFFICHER LONGUEUR( T) // 0
AFFICHER "(" & T & ")"  // ()

V <- " "
AFFICHER LONGUEUR( V) // 1
AFFICHER "(" & V & ")"  // ( )

S <- "Bonjour"
T <- "Bonjour"
SI S==T ALORS
        AFFICHER "VRAI"
FINSI

S <- "Bonjour"
T <- "Bonsoir"
SI S>T ALORS
        AFFICHER "FAUX"
FINSI

POUR I ALLANT DE 0 JUSQUA LONGUEUR( S) -1
        AFFICHER ORD( S[ I])
FINPOUR

66
111
110
106
111
```

```
117
114

// -> MAJUSCULE
POUR I ALLANT DE 0 JUSQUA LONGUEUR( S) -1
        SI ORD( S[ I]) >=97 ALORS
                S[ I ]= CHAR( ORD( S[ I]) -32))
        FINSI
FINPOUR
AFFICHER S              // BONJOUR

POUR I ALLANT DE 0 JUSQUA LONGUEUR( S) -1
        SI S[ I]=='o' ALORS
                S[ I] <- 'O'
        FINSI
FINPOUR
AFFICHER S      // BOnjOur
```

1 octet : 2^8 possibilité 0->256 אופציות

1 octet signé : -127 -> +128 מספרים בין

2 octets : 2^{16} -> 65535 -32768 -> +32767

i ← 0

Loops/boucles // etape

- While (tant que) כשהתנאי מתקיים - אז להכנס // - init : התחלה - increment
- Jusqu'à אני נכנס ללופ בלי תנאים, התנאי מוגדר בסוף // - test : לבחון - corps : גוף
- for int, test, inc בשורה אחת

Java SE Standard Edition -> = להוריד

-> JDK למתכנתים מכיל Compiler JRE inclus dans JDK

JRE אצל הקליינט -> כדי שיוכל להשתמש באפליקציה

-> Apache Netbeans -> Download All 221 MB

Java EE = בשביל ארגונים

בתיקיית JDK קיים קובץ Javac זה הקומפיילר

NetBeans

New project -> Java -> Java application

☆ Afficher -> System.out.println()

run file להפעלה MF6

צריך להעתיק את תיקיית הפרוייקט שנמצאת ב- C ולהעתיק אותה למחשב שלי

for	while
var local	var pas local

i אפשר להשתמש רק בלופ

Saisir ->

ctrl - רווח - השלמה אוטומטית

system.in →
in → a few sources מילה לועזית של המשתמש ← input
print ln → עובר לשורה הבאה   ln = line
print → (לא בסוף השורה)
next: ← זה אומר שאם אני רק מכניס מילה הוא עובר ל

Tableaux – Arrays

כשאני מגדיר טבלה אני צריך לדעת כמה פריטים יש בטבלה

```
int n=2
int H[] = new int [n]
```

אם אני משנה מאוחר יותר את n, הגודל של הטבלה לא משנה
ה-length הוא var - זה לא ככה ב-String

line en pseudocode ⇒ input
תן ליוזר להכניס משהו
אם יש לי Array של 25 פריטים אני לא יכול להוסיף 5
אני צריך ליצור טבלה חדשה של 30 פריטים, להעתיק את ה-25 הקיימים ולהוסיף עוד 5

String	Array	Algo	~~Otto~~	
14, 18-23	ok {8,12}	✓	דגמי מטלה	} 08/10

Chaînes de Charecters

'c' אות אחת
"chc" יותר מאות אחת
tableau : T[i]   T tableaux
chaine de charecter : s.chartAt(i)   S String
t = "_"
if (s.compareTo(t) → (s>t) >0   <0 (s<t)
האותיות הם מספרים. למשל A=75 (לא בדקתי) ולכן כשהוא משווה בין 2 הוא משווה מספרים
les Chaines → non modifiable

Algo

there are
- ✓ Sequential search
- ✓ Insertion sort
- Merege Sort
- ✓ Binary Sort

Sequential Sort

בעצם: יש רשימה של תלמידים ואני צריך למצוא את השם שלי בתוך הרשימה
תכלס: האלגוריתם עובר שם שם ברשימה ומשווה אותו למה שאני מחפש. אם הוא לא מוצא - הוא עובר לשם הבא ברשימה. הוא מפסיק כאשר: הרשימה נגמרת - כאשר השם נמצא

```
seq(list, name) -> input
length = □
found = false
index = 1
```
(length, found, index:) אני מספק 3 משתנים / משתנה מעקב (איפה אני)

```
while index ≤ length and found = false:
    if list[index] = name then
        found = True
    index = index + 1
```
- 2 תנאים להפסקת הלולאה
- אם מצאת את השם
- אז found ← True וזה מוציא מהלולאה
- כדי שאני אתקדם בלולאה

Insertion Sort

אבסטרקט: זה כמו שבמשחק קלפים אני מסדר את הקלפים לפי סדר עולה (ביחס לקלף הכי שמאלי)
תכלס: כל פעם שהאלגוריתם קורא מספר (קלף) הוא מסדר וממקם אותו ביחד עם המספרים שכבר מיינו
ההתייחסות היא ל-2 קבוצות שונות: קבוצת הלא שכבר מיינו וקבוצה של לא מיינו ולכן צריך לעקוב
אחר שניהם באמצעות 2 משתני מעקב. אני מתחיל מקבוצה אחת אבל על ההתחלה האלגוריתם מתייחס
לקלף הראשון כקבוצה בפני עצמה (מספרים שמוינו כבר). הקבוצה השני זה כל שאר הקלפים
ככל שהאלגוריתם מתקדם -> קלפים עוברים מקבוצת הלא ממוינים לקבוצת הממוינים. האלגוריתם משווה כל מס' ->
למס' הכי ימני בקבוצת הממוינים. אם הוא קטן יותר -> האלגוריתם מזיז את כל המספרים מקום אחד וממשיך להשוות
סיום: כשה-list unsorted ריקה
יעילות: מתאימה קטנות או רשימות שמוינו בעבר (והוסיפו להם פרטים עלי להוציא למיין אותם)

```
insertion(list)
length =       // מס' הפריטים ברשימה
n_index = 2          // האלמנט השני שמסמן את תחילתו של Unsorted      (מיקום ה- new)
while n_index ≤ length          // סיים כאשר עברנו על כל הרשימה
new = list[n_index]        // המספר שאותו אני בוחנים. (עכשיו) ב- Unsorted
S_index = n_index-1
While new < list[s_index]    // מספר שאותו כבר בחנתי > המספר שאותו אני בוחן
                AND   S_index > 0         //   S_index=0 ⇐ ב-min
list[s_index+1] = list[s_index]       // תקדם אותו צעד ימינה
s_index = s_index-1        // לבדוק גם את המספר צעד שמאלה
list[s_index+1] = new       // הוסף המספר שנבחן במקום הריק ל- new
```

עוד דרכים לכתוב:

```
list[          ]
i ← 1
while i < length(list)
x ← list[i]
j ← i-1
while j ≥ 0 and list[j] x >
    list[j+1] ← list[j]
    j ← j-1
  End While
  list[j+1] ← x
  i ← i+1
End While
```

```
i ← 1
while i < length(list)
j ← i

while j > 0 and list[j-1] > list[j]
  swap list[j] and list[j-1]
  j ← j-1
  End while

  i ← i+1
  End While
```

Binary Search

הוא כמו ה- seq search אבל הרבה יותר יעיל ומהיר. אם מיליון צעדים ב- seq לעומת 20 בbinary

תנאי: מדובר ברשימה שמסודרת ← ordered

רעיון: כל הזמן מחלקים את הרשימה בחצי. משווים את ה- mid item במרכז למה שאני מחפש.

אם האלמנט שאותו אני מחפש קטן יותר ← האלגוריתם קורא את האלמנט שנמצא באמצע החצי הראשון

אם האלמנט גדול יותר ← האלגוריתם קורא את האלמנט שנמצא באמצע החצי השני

```
Binary (list, item)
begin = 1
end = length    // אורך הרשימה
found = false

    While found = false And begin ≤ end :
    mid = (begin + end) / 2
    if list [mid] == item :
    found = True
    Si item < list [mid]
     end = mid - 1
     else :
     begin = mid + 1
```

מחפש Y

Δ = האמצע של...

A, B, ... | T, ... X, Y, Z

האם Y גדול מ-Δ או קטן (או שווה)

← גדול

... T ... | X, Y, Z

האם Y גדול מ-Δ או קטן (או שווה)

← גדול

X, Y, Z

האם Y ... או שווה ל-Δ ?

← שווה

Meregl Sort

שימוש בפרדיגמת recursive: חיתוך הבעיה לחתיכות קטנות יותר ואז הרכבה מחדש

הרכבה מחדש: לקיחת 2 סלים של מספרים ממוינים ולמזג אותם לסל אחד

אנלוגיה: אפשר לדמיין 2 חפיסות קלפים שכל אחת מהם ממוינת מהקטן לגדול. כל הקלפים הפוכים מלבד הקלף הכי קטן בכל ערימה. האלגוריתם משווה את 2 הקלפים הגלויים לעין. את הקטן מביניהם הוא מוציא ומניח אותו בערימה חדשה. ואז קלף חדש נחשף ושוב משווים לקלף שנותר חשוף וכך הלאה..

האלגוריתם ממשיך עד אשר כל הקלפים הונחו בערימה החדשה

כאשר סיימתי סל אחד של קלפים ובסל השני נשארו לי קלפים - אני צריך להעביר את כל הקלפים שנשארו לערימה ה-3.

mereg(listA, listB) → output listC

index ← 1 // מעקב אחר הפריט בערימה C

איך לכתוב אלגוריתם?

ב-6 שלבים

1. what is the outcome?

הגדרת היעד, אם אני לא אדע לאן אני רוצה להגיע ← לא אוכל להגיע לשם

2. Start point

מהי נקודת ההתחלה ← נתון: איזה data / input נתונים לי, איפה זה נמצא

← אילו חוקים ומושגים רלוונטיים עבור ה data הזה

← איך הערכים של ה- data קשורים אחד לשני

3. Ending point

מהי נקודת הסוף ← מה נראה כתוצאה מהתהליך

← מהם השינויים? מה יעלם? מה יתווסף?

4. list the steps

5. How to accomplish each step

פירוק ה- step למרכיבים קטנים יותר בשפת Java

6. Review

האם האלגוריתם פתר את הבעיה/השיג את היעד?

האם ברור ומוגדר מה ה- input/ output

האם אפשר לפשט את כל אחד מהצעדים?

האם בוודאות האלגוריתם יסתיים עם התוצאה הנכונה?

ב-3 שלבים

- זיהוי ה-:

• תהליך • ההחלטות העיקריות • לופים • משתנים

- אלגוריתם:

- כתיבת אלגוריתם - walk through באלגוריתם, צעד צעד. האם נתקלת בבעיות?

Fine-Tuning:

- תיקון בעיות שנמצאו - קבל תהליכים - קבל משתנים - test

```
8       System.out.println("b = " + b);
9       float c = random.nextFloat();
10      System.out.println("c = " + c);
11      if (a < b)
12        if (b < c) System.out.println("a < b < c");
13        else // a < b et b >= c
14           if (a < c) System.out.println("a < c <= b");
15           else System.out.println("c < a <= b");
16       else // a >= b
17          if (a < c) System.out.println("b <= a < c");
18          else // a >= b et a >= c
19             if (b < c) System.out.println("b < c <= a");
20             else System.out.println("c <= b <= a");
21    }
22 }
```

Soit a est inférieur à b, soit il ne l'est pas. S'il est inférieur, il reste trois alternatives à considérer : $a < b < c$, $a < c < b$ ou $c < a < b$.

Les conditionnelles des lignes 12 et 14 vous permettent de tester ces possibilités. De la même manière, les trois alternatives de a supérieur à b sont testées par les conditions des lignes 17 et 19. La figure 3.2 représente l'arbre décisionnel des six résultats possibles de ce programme.

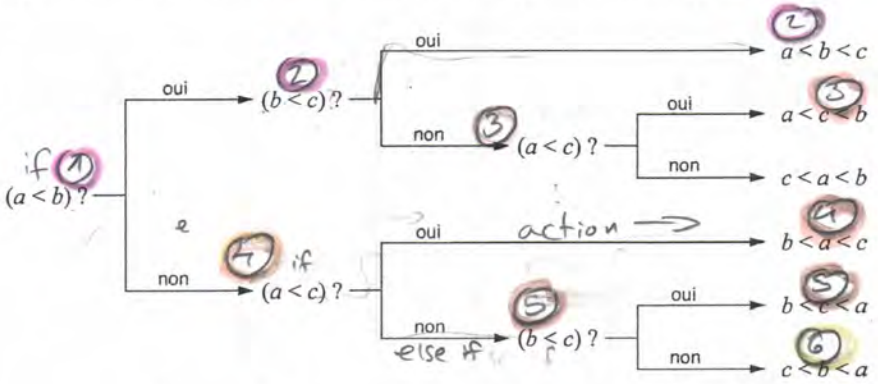


Figure 3.2 Arbre décisionnel de l'exemple 3.4

Voici la sortie d'une exécution :

```
a = 0.34565938
b = 0.3545665
c = 0.057765722
c < a < b
```

Notez comment le contenu des instructions `if..else...` est indenté à l'exemple 3.4. Si vous prenez la peine de formater correctement votre code sur ce modèle, vous comprendrez plus aisément la logique de ce type de structure conditionnelle et faciliterez ainsi leur utilisation par ceux qui lisent votre code Java. Mais restez vigilant car ces structures sont à l'origine des pires bogues, à savoir les erreurs de logi-

Algo - Java

```
public class algo01 {

    public static void main(String[] args) {

        System.out.println("Hello, world !");

        int l;
        l = 0;
        System.out.println(l);

        int m = 20;
        int n = 30;

        l = m + n;
        System.out.println(l);
        l = m - n;
        System.out.println(l);
        l = m * n;
        System.out.println(l);
        l = m / n;
        System.out.println(l);

        if (m > n) {
            System.out.println("m > n");
        } else {
            System.out.println("m <= n");
        }
////////////////////////////////////////////////////////////////////////////
        int k = 0;
        while (k < 5) {
            k = k + 1;  // ++k ou k++ ou k+=1
            System.out.println(k);
        }
////////////////////////////////////////////////////////////////////////////
        int j = 0;
        do {
            j++;
            System.out.println(j);
        } while (j < 5);
////////////////////////////////////////////////////////////////////////////
        for (int i = 0; i < 5; i++) {
            System.out.println(i);
        }

    }

}
```

input

```
import java.util.Scanner;

public class saisie {

    public static void main(String[] args) {

        Scanner clavier= new Scanner( System.in); // create a scanner object
        System.out.print( "Saisir une chaine : ");
        String s= clavier.next();
        System.out.println( s);

        System.out.print( "Saisir un nombre : ");
        int i= clavier.nextInt();
        System.out.println( i);

    }

}
```

var

input

reading

output

int

print

assign the input to i variable

1. creating object (scanner)
2. Asking the user for information (System.out)
3. Reading the input (String/int)

```
int no
for (                    ) {
  System.out.print ("Enter...")
  no = input.nextInt
```

var

Array

```
import java.util.Scanner;

public class tableaux {

    public static void main(String[] args) {

        int[] t;
        t = new int[10];

        int n = 20;
        int tt[] = new int[n];

        int[] ttt = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

        System.out.println(ttt.length);

        for (int i = 0; i < ttt.length; i++) {
            System.out.println(ttt[i]);
        }

        Scanner clavier = new Scanner(System.in);
        for (int i = 0; i < t.length; ++i) {
            t[i] = clavier.nextInt();
        }

        System.out.println("---");
        for (int i = 0; i < t.length; i++) {
            System.out.println(t[i]);
        }

    }

}
```

String

```
public class chaine {

    public static void main(String[] args) {

        String s;
        s = "Bonjour";
        System.out.println( s);
        System.out.println( s.length());
        String t = "Comment ça va ?";

        String v= "toto";
        v = s + ", " + t;
        System.out.println( v);
        System.out.println( v.length());

        s = "Hello";
        for( int i = 0; i< s.length(); i++)
            System.out.println( i+':'+s.charAt(i));

//        System.out.println( s.charAt( 5));

        t = "";
        System.out.println( t.length());
        System.out.println( "(" + t + ")");

        t = " ";
        System.out.println( t.length());
        System.out.println( "(" + t + ")");

        s = "Bonjour";
        t = "Bon";
        t = t +"jour";
        System.out.println(s);
        System.out.println(t);
//        if( s==t)     // adresse, handle
        if( s.equals(t))
            System.out.println("s est égal à t");
        else
            System.out.println("s n'est pas égal à t");

        t= "Bonsoir";
        if( s.compareTo(t) > 0)     // <0  ==0 >0
            System.out.println("s > t");
        else
            System.out.println(" s <= t");


    }

}
```

Methods / Functions

```
import java.util.Scanner;

public class structure {

    static int saisirInt(String prompt) {
        Scanner clavier = new Scanner(System.in);
        System.out.print(prompt);
        return clavier.nextInt();
    }

    static String saisirString(String prompt) {
        Scanner clavier = new Scanner(System.in);
        System.out.print(prompt);
        return clavier.nextLine();
    }

    static void afficher(int n) {
        for (int i = 0; i < n; i++) {
            System.out.println(i);
        }
//        n = 20;
//        System.out.println("afficher : " + n);
    }

    static void saisirTableau(int[] t) {
        for (int i = 0; i < t.length; i++) {
            t[i] = saisirInt("Saisir t[" + i + "] : ");
        }
    }

    static void afficherTableau(int[] tt) {
        for (int i = 0; i < tt.length; i++) {
            System.out.println(i + ":" + tt[i]);
        }
    }

    public static void main(String[] args) {

        int n = saisirInt("Saisir un n : ");
//        int m= saisirInt("Saisir un m : ");
//        saisirInt("Saisir un toto : ");
        System.out.println("n=" + n);
        afficher(n);
//        System.out.println("après afficher : "+ n);
        int tableau[] = new int[n];
        saisirTableau(tableau);
        afficherTableau(tableau);
//        String nom= saisirString("Saisir un nom : ");
//        System.out.println("Nom : "+ nom);
    }

}
```

static type name ( ) { }

static void afficher ( ) { }

numero

Programmation structurée

rupture: boucle
test

function = method = Block

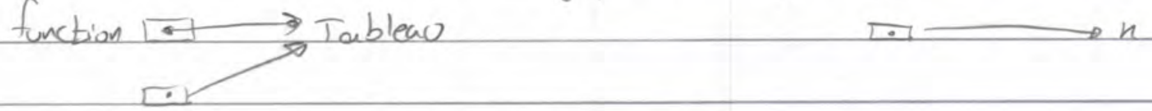
functio — return : return information
\ ≠ return : fait un action (void)

variable = function (return de informatio et le mettre en var.)
chaque method a une signature: static/pas, argument,
return ⇐, אם אין ~~קורא~~ קורא לפונקציה אך אין לו ארגומנטים לה var של היא תקח את ה- info
אבל תצטרך לצבור כי אין לה var

~~static~~ void – return rien
method → on travaille sur un copy (local)

static int []
↑ return

איך זה נשמר בזכרון?



לא עושה copy של ה- Array
← עושה copy של ה- Adress
ומה? במקום שלה Object

Metis

metis.afpa.fr user: 20060356
סיסמא: Afpa 20060356!
סיסמא: Kipod51G@

16 נובמבר אנחנו בחופש מ-Afpa וגם ב-2 דצמבר
החלק של Merise ; SQL באתר "יוסף" החל מה-30 לאוקטובר

Merise

10

faire boger base de donne — אם אני צריך לבנות דבר מסודר ← Merise
מכיל Vocabulaire מיוחד

Analyse ← Model Design ← מודל לעבודה על נייר (דף) - לא חייב תוכנה ← SQL
construire le chemain physique → SQL

מקבילה ל- Merise : Gama.

Base de donné

1. tous les 🗎 → טפסים
2. Analyse: extract of the מידע הדרוש (qui vont construire la base de donné)
3. Dictionner de donné: code nom de stagaire → NomStagaire
   - + Description → the name of the stagaire
   - + Type → int, string
   - + Range (טווח) : range of the values → >0, Man\Woman
   - + Caculation rules

entités — הקשר בינהם — Niveau Conceptuel — M.C.D

Niveau Organisationnel — M.O.D - M.L.D (logique)

SGF, SGBD — Niveau Physique — M.P.D

Conceptuel : Dictionnaire des donnes

1. איסוף מידע: מסמכים, טפסים. מה צריך להכניס ל- BD
2. יצירת רשימה של כל הנתונים
(*. אם ה- BD כבר קיים לבדוק איזה נתונים אני צריך להוסיף שלא קיימים)
3. לשים בצד את הנתונים שהם תוצאה של חישוב
4. למיין הבדיקה Synonymes : נתון אחד עם שמות שונים
homonymes : שני (או יותר) נתונים שנמצאים תחת שם אחד וצריך לפצל
redondance : חזרות

Dictionnaire

N°	Nom	Code	type( )	Observation

תוכנות לשלב Conceptuel (שמתאימות לכל ...)

* PowerAMC   * MySQL Workbench   * DBDesigner

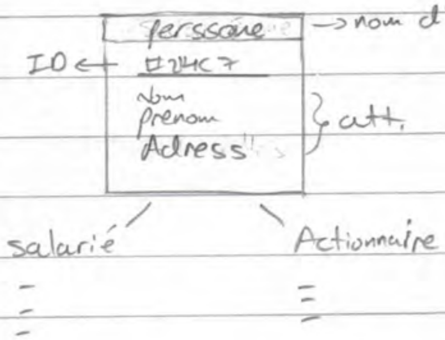
M.C.D

entité: ensemble des objects du meme nature ex: perssone class

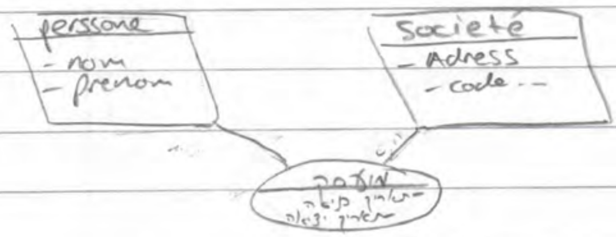
Occurence d'entité: individu ex: pini

Attribut: information que on peux attacher à une entité → 1 valeur! sex: Homme

Identifiant: att particulier qui permet d'accéder à un occurrence precis #214C7



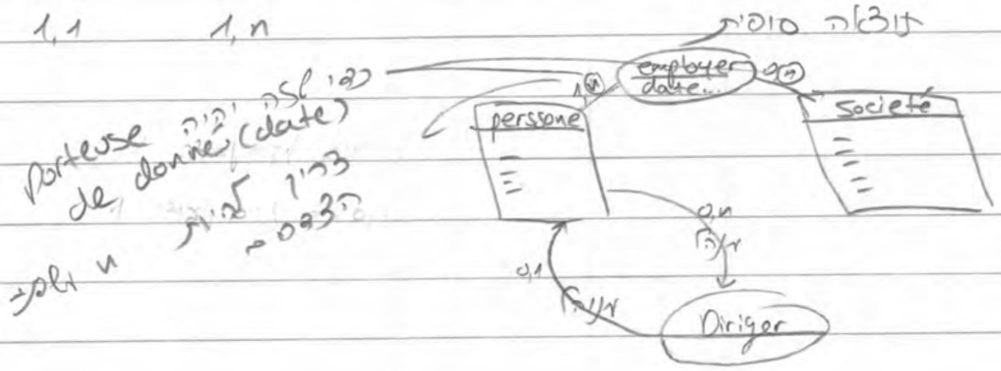
RELATIONS

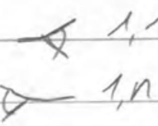


CARDINALS

0,1 0,n

1,1 1,n





Data Integrity

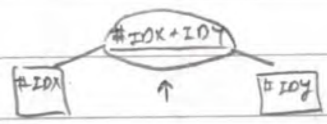
\# valeur: Bush George 0617894 != Bush Georg 0617897 → ID

\#ref: soulation (0,1) (1,1)

\# domaine: range, min, max ex: salaire ≥ 1300€

\# null: x != null

Identifiant:



ID - 2

MCD סיכום

1. בניית Dictionnaire de donnée
2. לנקות ← Synonme, homonyme ונתונים חסרי משמעות
3. להוסיף כאלה שנראים הכרחיים
4. זיהוי ה- entites הברורים לעין
5. הצגתם: entite + id + מאפיינים
6. חיבור ה- entites ביניהם
7. Cardinal: כלל - כמה פעמים לכל הפחות (מינימום) ולכל היותר (מקסימום) לוקחת חלק ה- entite בסיטואציה הנ"ל
8. וידוא סופי: כל הנתונים משלב 1 נמצאים
   - לכל entite יש לפחות מאפיין id אחד
   - כל ה- cardinal במקום
   - כל הכללים של העסק מיושמים
   - כל אינפורמציה עתידית של הנתונים תהיה בתוך השלב בהקצאת נייר וגלויה בקצוציות בין ה- entites

Collection

Ctrl espace

(java) ArrayList ~ vector (Swing)

sout

- HashMap : clé par object

(sout) System.. print

for (type nom.value : array.name)

Arraylist<type> a = new Arraylist()

Collection name = new Arraylist(); /Arraylist nom =

a.clear()

a.is.empty — true / false

a.remove

a.size()

- toArray → collection tranform into array

a.sort()

get(int index) → value that in position int

```
for(object s:a)
  if (s instanceof Integer)
    a.remove(s)
```

1. create a new array
2. copy all the object that I want to delete a_2
3. create a boucle that delete all the objects from a_2 that are found in a_1

le mechanichem déaces de tableux est plus stable

Arraylist

Array

collections : bibilo de functionality

collections.sort();

sort mix types →

- compareto pour trier

game: Mass Effect

compare

String have inside it the compareto

Objects doesn't have it. I need to implemtnts the Interface Comparable

-> We need to define it in the object's class (like with toStrings)

<0 = the other is bigger

0 = egal

>0 = what I compare is bigger

```
this.nom.compareTo(o.getnom());
```

אני צריכה לדרוס את המתודה compareto ←

```
// (stagaire)a; -> cast
```
[illegible]

```
// class comparator -> recive 2 object
// comparable       -> recive 1 object
```

Collection

```
import java.util.ArrayList;
import java.util.Collection;

public class mesArrayList {

    public static void main(String[] args) {

        ArrayList a = new ArrayList();

        a.add("aa");
        a.add("bb");
        a.add("cc");
//        System.out.println( a.size()+"/"+a.isEmpty());
//        a.clear();
//        System.out.println( a.size()+"/"+a.isEmpty());
        a.add("dd");
        a.add("ee");
        a.add(2, 123);

        System.out.println(a.size());
        for (Object s : a) {
            System.out.println(s);
        }
/*
        System.out.println("++++");
        ArrayList aa= new ArrayList();
        for (Object s : a) {
            System.out.println(s);
            if (s instanceof String) {
                aa.add(s);
            }
        }
        System.out.println(a);
        for (Object s : aa) {
            a.remove( s);
        }
        aa.clear();
        aa= null;
*/
        System.out.println(a);
//        a.remove(3);
        for (int i = 0; i < a.size(); i++) {
            if (i == 2) {
                a.remove(i);
            }
            System.out.println(i + ":" + a.get(i));
        }

    }

}
```

Collection

```
package classes;

public class Stagiaire implements Comparable<Stagiaire>{

    private String nom;
    private String prenom;
    private String id;

    public Stagiaire(String nom, String prenom, String id) {
        this.nom = nom;
        this.prenom = prenom;
        this.id = id;
    }

    public String getNom() {
        return nom;
    }

    public String getPrenom() {
        return prenom;
    }

    public String getId() {
        return id;
    }

    @Override
    public String toString() {
        return nom + " " + prenom + ", id=" + id;
    }

    @Override
    public int compareTo(Stagiaire o) {
        if(  this.nom.equals(o.getNom()))
            return this.prenom.compareTo( o.getPrenom());
        return this.nom.compareTo( o.getNom());
    }
}
```

1 argument

another class

```
package classes;

import java.util.Comparator;

public class prenomCompare implements Comparator {

    @Override
```

2 arguments

<stagiaire>

cast

```
    public int compare(Object a, Object b) {
        Stagiaire aa= (Stagiaire) a;
        Stagiaire bb= (Stagiaire) b;

        return aa.getPrenom().compareTo( bb.getPrenom());
    }

}
```

Exceptions

1) create class
2) extends Exception → generic

s.trim ⇒ delete all the spaces before/or after

Class test

method x() throws ______ name of the exp class (↑ a must if it's a checked exception) אני יוצר method

if ... (something happens) → throw new ______ ()

import exceptions. name of the exp class

if I put throws ______ in the main method זה נותן את זה לתוכנה ולא למשתמש

Class Main

class.method();

try{ class.metod; }
catch(   ) {   מה לעשות }

בהגדרה ל- Exception Class

→ constructor : I create a constructor for the message

אחרי שיצרתי את ה-constructor שלקבל גם message אני יכול לחזור ל- class test ולכתוב את ההודעה של ה- exc כמו שאני רוצה, אני שם אותה ב- ()

Override Java Exception

הבעיה של Excep שקיים כבר ב-java שאני רוצה להתאים אותו לצרכים שלי, שינוי את ההודעה שאני רוצה וכו'..

1. להריץ את הקוד עם "הטעות" ⇐ קבלת except ⇐ להסתכל, מה השם של ה-excep?

: java.lang.NumberFormatException

catch(______)

או

אבל אני לא חייב! אם אני חושש זה עובד טוב!   catch (Exception e)

אם בתוך method (חוץ מ- main) שאני כותב בו try...catch אני לא חייב להוסיף throws בתוך החתימה של ה- method

זה שאני יכול לשים אחרי ה- catch   exceptionname.printStackTrace();

עם המלצה ל- מהו ה- erreur בקונסול java, איזה שורה וכו'..

Class exception extends Exception

אני יכול להוסיף בנוסף ל- message פונקציה נוספת: להוסיף את קוד ה - erreur
בנוסף להודעה

1) Declare private int ErrorCode
2) Add a constructor with the ErrorCode (that accepts ...)
3) Add a method getErrorCode

Run Time Exceptions
class name extends RuntimeException {
....
....

A method that use a RunTime Exception doesn't have throws

דוגמה לבעיות RunTime קורה ב- input
הלא כל input שלא מתאים מהימן ⇐ run Time
כמו user ⇐ זה runTime זה
כל השגיאה הוא facultative זה אז כלום כמו זה לא Exception Obligatoire
זה לא RunTime Exp.
מוסר כל כל השגיאה הוא הכרחי, למשל (prenom, nom) זה ב- exp Obligatoire

the Compiler doesn't looks for/search for runtime exp

the Compiler
- check → Exception
  MUST: try{ }chatch{ }
- doesn't ~~don't~~ check → runtime exception
  I can take a risk or not (try...)
  and the risky method doesn't have to "throws"

Exceptions

```
package classes;

import exceptions.mesExceptions;
import exceptions.mesRTExceptions;

public class mesOutils {

    static public String setNom( String s)
            throws mesExceptions {

        if( s==null)
            throw new mesExceptions(1,"Le nom ne peut être null !");
        if( s.trim().isEmpty())
            throw new mesExceptions("Le nom doit contenir des caractères !");
        if( s.trim().length()<1)
            throw new mesExceptions(3,"Le nom doit comporter 1 caractère !");
        return s.trim().toUpperCase();
    }
    static public String setPrenom( String s) {
        if( s==null)
            throw new mesRTExceptions(10, "Le prenom ne peut être null !");
        if( s.trim().isEmpty())
            throw new mesRTExceptions(11, "Le prenom ne peut être vide !");
        return s.trim().toLowerCase();
    }

    static public int getRef( String s) throws mesExceptions {

        try {
            int j= Integer.parseInt(s);
            return j;
        } catch (NumberFormatException e) {
            // e.printStackTrace();
            throw new mesExceptions(4,"Valeur non numérique !");
        }

    }

}
```

after throw there is new

```
import classes.mesOutils;
import exceptions.mesExceptions;
import exceptions.mesRTExceptions;
import java.util.logging.Level;
import java.util.logging.Logger;

public class testEx {

    public static void main(String[] args) {

        try {
            mesOutils.setNom(null);
        } catch (mesExceptions ex) {
            System.out.println("Oops:" + ex.getMessage());
        }
        try {
            mesOutils.setNom("     ");
        } catch (mesExceptions ex) {
            System.out.println("Ooops:" + ex.getMessage());
        }
        try {
            mesOutils.setNom("");
        } catch (mesExceptions ex) {
            System.out.println("Oooops:" + ex.getErrorCode() + "/" +
ex.getMessage());
        }
        try {
            mesOutils.setNom("o");
        } catch (mesExceptions ex) {
            System.out.println("Ooooops:" + ex.getMessage());
        }
        try {
            String s = mesOutils.setNom("toto");
            System.out.println(s);
        } catch (mesExceptions ex) {
            System.out.println("Ooooooops:" + ex.getMessage());
        }
        try {
            int j = mesOutils.getRef("123");
            System.out.println(j);
        } catch (mesExceptions ex) {
            System.out.println("Oops:" + ex.getMessage());
        }
        try {
            mesOutils.getRef("abc");
        } catch (mesExceptions ex) {
            // ex.printStackTrace();
            System.out.println("Oops:" + ex.getMessage());
        }

        String s = mesOutils.setPrenom("Toto");
        System.out.println(s);
        try {
```

Exceptions

```
            mesOutils.setPrenom("    ");
        } catch (mesRTExceptions e) {
            System.out.println("Oops:" + e.getMessage());
        }
        mesOutils.setPrenom(null);

        System.out.println("Done!");

    }

}
```

Exceptions

```
package exceptions;

public class mesExceptions extends Exception {

    private int ErrorCode;

    public mesExceptions() {
    }

    public mesExceptions(String message) {
        super(message);
    }
    public mesExceptions(int ErrorCode, String message) {
        super(message);
        this.ErrorCode=  ErrorCode;
    }

    public int getErrorCode() {
        return ErrorCode;
    }

}
```

```
package exceptions;

public class mesRTExceptions extends RuntimeException {

    private int ErrorCode;

    public mesRTExceptions() {
    }

    public mesRTExceptions(String message) {
        super(message);
    }

    public mesRTExceptions(int ErrorCode, String message) {
        super(message);
        this.ErrorCode = ErrorCode;
    }

    public int getErrorCode() {
        return ErrorCode;
    }

}
```

SQL

Amine

Donnè : brut, sans sens
information : donnè + sens
SGBD - ex: studio managment

DDL : Create, Alter, Drop
DML : Select, Insert, Delete, Update
DCL : Data Control Language : Grant, Revoke — מתן גישה למשתמשים

מסבה הכל
↓ Group By
Having
Order by

Count(*) — סופר את כל השורות
count(column) — תמיד אחרי count יהיה ‖ ולא ⇐

PRINT GETDATE() — מציג את ה- date של היום

Carecters
char(x) — את המקומות הריקים הוא ממלא ברווחים
char(8)
A M I N E _ _ _
select * where nom='AMINE' → it wouldn't find it cause AMINE doesn't exist! but 'AMINE___' exist

PL SQL — שפה שמאפשרת לעשות לופים וכאלה..

Having — במקרה שיש לי פונקציה של aggregation אני חייב להשתמש ב
ORACLE ⇒ SQL Developer

Base de Donné

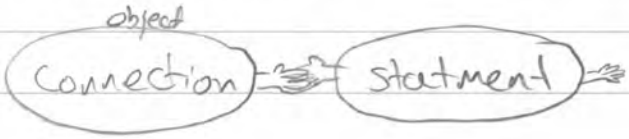
Amine

- ① telecharger le JDBC (pilote) (JDBC SQL Server)
  Eclipes (right click) → Build path → configure build path
  chosse the JAR that corresp of my verison

String connection = base de donne path
Create a connection Object
Create a stm Object
String requet = " "
ResultSet
MetaData

Insert → excuteUpdate (queryVariableName) → return the no of modified lines
Select → excuteQuery (Select Only)

Setting SQL
TCP/IP activated



2eme Project
2 types dutilisateurs 1: afficher (select)
2: adminsitrateur (tous faire)

login { create table → login email → cle primaire 1st field
password 2st
admin/user niveau dacces 3

Admin → add
→ gestion de vente

Afficher les commande

JDBC

1) connect to SQL 2) SQL 3) copy connection code 4) paste in Java
5) add Library 6) connect 7) insert 8) colse connection

connect: users - premissions - acess (words norme JDBC)
the connection is diffrent from 1 to another (mysqu vs Oracle vs SQL...)
the script can be found in the MS Documentation.

Java

main class → copy paste code
after try → instruction to charge the driver that one I copy to Java
→ Erreur → right click (catch... try)

class.forName → Driver upload-ו
* Add Library → I am looking for a Driver SQL isn't there so...
→ create → giving it a name (MSSQL Driver) → Add Jar (sqljdbc 7.2 → msq... 8/11)
(if I have version 8 → 8 if 11 → 11) → Add Library (ok)
* Download MS SQL Driver
▷ → no errors after importing the Library
// create a variable for...
copy to the main
after the connection var
Fix imports → verifay java.sql.connection
The procces of connection is a risky → try... catch => System.out.print("Done")
Localhost:1433; → port
"databasename=change the name of the DB;" // DB
"user: sa; password: sa"

```
statement var = connection.createStatement()   => risk → try_catch
String query = "בקשה"
```

→ excucate.update(), return int → number of lines that been affected)

```
int resulat = stat.excuteUpdate(query)
System.out("you succed to insert" + resulat + "lines")
```

try catch with SQL

xxx.getErrorCode() → only in SQL

ex : 2676 → (in java) if erreur = 2676 so...

last step

connexion.close() → risk try catch

Statsments are also small connections so I need to close it also

Stsm.close()

new main class (זכור: SELECT כמו, INSERT-ו CREATE כמו 36

SELECT

כאן הוא הכלה שלנו צריך לאחזר, לזכור connection ל-DB. הקיה יותר מסודר

after the connection has been established... we create statments

the object ResultSet have get method For all the types of SQL

ResultSet rs = Stmt.execute Query(query)

.next() → return true if he find false if not קורא את ה-cols השורות אחת אחת

While (rs.next()) loop ⇐ קורא שורה שורה עד שאין עוד

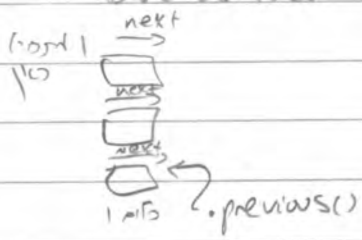
ex: rs.getINT(1) ↑ no. index can be replaced by the column's name

rs.getINt("ref")

SQL שומר הכל בתור text ואחר כך ממיר ל- float, int וכו'...

rs.get String("ref") → it will work because it was born in SQL as text

I have arrived to the end



.creatStatment() doesn't allow the use previous()

create Statment (, concur_...)

to enable previous (

PrepareStatment → Security

ברגע שיש לי request שקשורה ב-input של user הם פורצים, אך לא משתמש ב-Statment
אני משתמש ב-prepareStatment

```
PrepareStatment pstm = connexion.prepareStatment (query);
pstmt.setInt (1, valeur)      ex → setInt (1, 6.0)
```

(index above "1"; arrow from pstmt to:) 1 for each column

```
int result = pstmt.executeUpdate ( );
```

→ forbiddent to put something here (refers to the brackets)

חיסרון: not possible to see the final result before sending

BD + Java

import : select

export : insert

Hash Map

for (:   ) doesn't work cause there isn't iterator

for (map.values( )) → work

אם יש בעיית התחברות לשרת SQL

cmd → mmc.exe

File → Ajouter / supprimer → Ajouter → SQL config (before the last one) manager

go to the main window → Select it → Config reseaux SQL Server

→ protocole TCP/IP activé → proporties adress → TCP 1433

→ adress dynamique → null

Java Regular Expression Regex

A regular expression is a sequence of characters that form a search pattern. For ex: only numbers
when you search inside a text you can use this search pattern to describe what you are searching for. it can be a single char or more complex pattern
they can do all types of search / replace

Import
java.util.regex. has 3 classes: Pattern
Matcher
PatternSyntaxException

I need to use both of the classes to create 2 objects

① Pattern pattern = Pattern.compile("Hello", Pattern.CASE_INSENSATIVE);
Matcher matcher = pattern.matcher("Hello My friend");

boolean matchFound = match.find() ->

if (matchFound) → if true // do that
else // do this

find(): true if the pattern was found
false → wasn't found

Pattern.compile( )
[ ] → searching in the range ex [0-9] [abc]
[^] → not between the range
| → any of ex cat|dog|horse
. just one instance of every character
^ find a match as the begining of the string $ at the end
\d find a number
\s find a whitespace
\b at the begining of the word \bword or at the end Word\b

+ one or more times

* zero or more

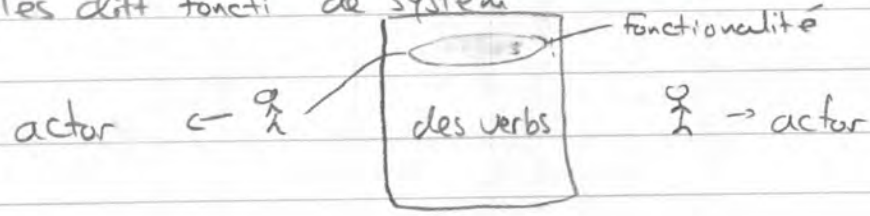
\d* would match (return true) an empty string

\d+ not

UML

Diagram de use case

presente les diff foncti de system



Include —

אם פעולה לא יכולה להתקיים בלי פעולה אחרת אז היחסים ביניהם הם יחסים של include

לדוג
retiner l'argent —— recevoir des argent

אם אין יותר את X, אין יותר גם את Y

Extend -----

כאשר פעולה אחת לא חייבת את קיומה של פעולה אחרת, אלא יחסים של אולי. אולי כן, אולי לא.

ex : payer-----recevoir une facture

אם אין יותר את X, Y עדיין יתקיים

Specilization ←

כאשר יש כמה אפשרויות לאותו nature. הרעיון: הם partager אותו סוג לדוג:

payer des articles ← payer en espace

← payer en check ...

Les Fonctionalité sont toujours des verbs

- MOA : responsable de comprendre la fonctionalité d'une system
- MOE :

Diagram de class

present la structure statique de system

nom de clas	regroupment d'info dans un entité logique
attributs	une characheristique de cette entité
methods	les actions que la class peut faire

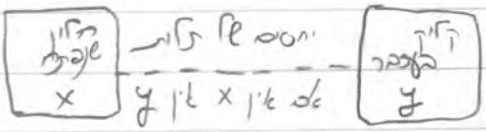
\+ public : visible par toutes les class

\# private : att/method accesible que depuis l'interieur de la meme classe

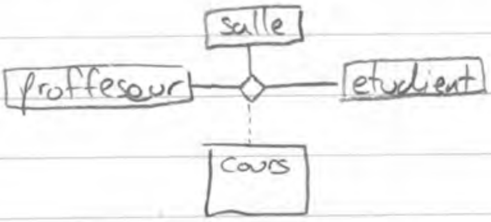
\- protected : accesible pour tous les class dans la meme packadge ou les class qui le herite → heritage/meme class

Cardinalité

Relation de dependence : designé avec les - - - - -



Class d'association many — many

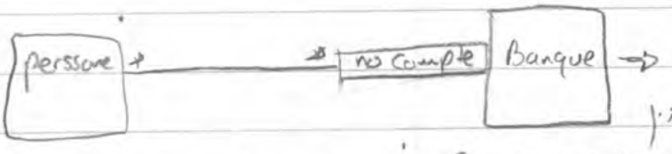


CRUD methods צריך שיהיה לנו class תמיד בכל

Create Read Update Delete

LDD → Drop, create, ALter

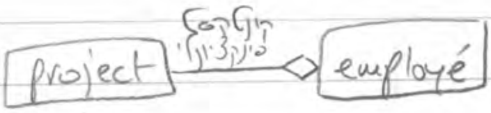
LMD → Delete, update



זה בעצם אומר ה- class הוא בנק.
גם לו הייתי מעצב את זה ככה אז היה אפשר להבין
שבעצם יכול להיות בעצם כמה בנקים אבל אני רוצה יותר כמה חשבונות בנק באותו הבנק.

Agregation אין תלות

une relation de subordination

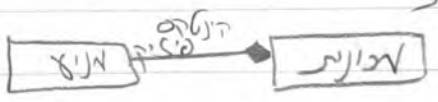


אם אני מוחק את ה- project זה לא מוחק את ה- employé
אם אני מוחק את ה- employé זה לא מוחק את ה- project

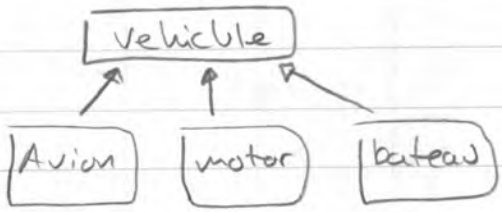
l'element agregé peut etre partagé
contance fonctionelle

Composition אם אני רוצה להכניס לדוגמה של תלות dependance אז יוצר תת-סוג שנקרא

Composition



Speccetibation \ Centralisation Heritage



Packages

- element dorganisation .

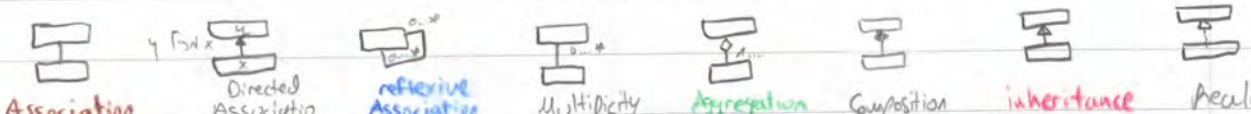
—▷ class of this packadge herite another class in pacadge 2

----▷ at least 1 class of that puck use the service of at least 1 class of the other packadge

- 1 pack can have sous-packdge

Class Diagrame

Relationship



Aggregation: the containd class are not strongly dependent of the lifecycle of the container

- Composition: like aggregation but the containd class is dstr if the container class is destroyed

Sequence Diagram - Sequence of events

Show how obj in the system or classes in our code interect with each other unlike a user case Dgm - this one is dynamique with a "time" sense

שלב 1

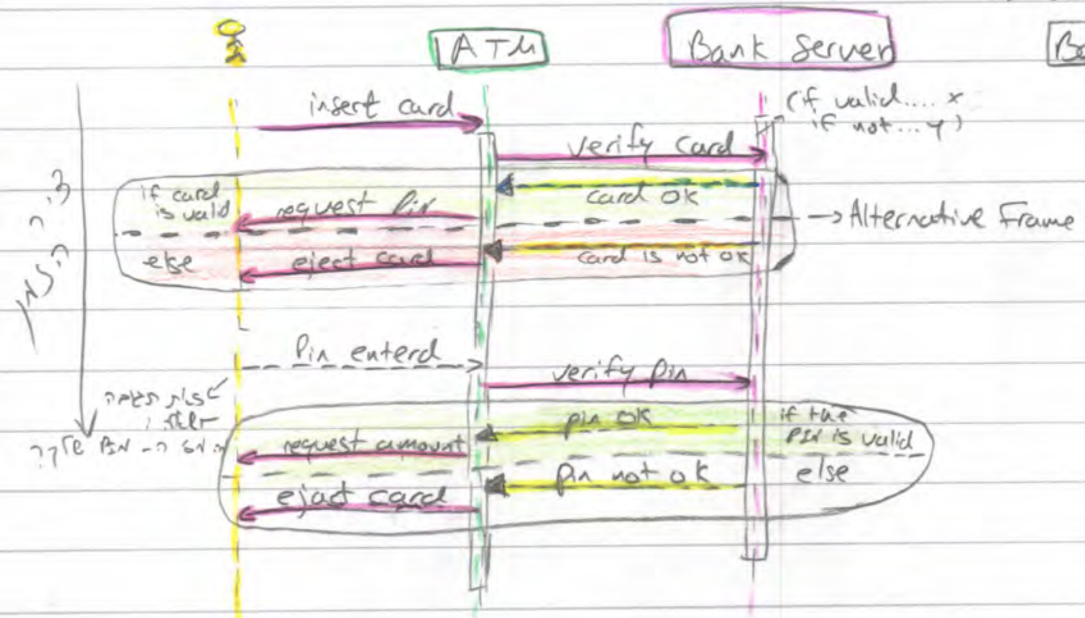
נזהה את כל המשתתפים: אם מדובר בקוד ⇐ כל ה-class אם מדובר ב-system ⇐ כל החלקים שלה

נגיד מערכת של ATM:

□ אובייקטים מוצגים ע"י ⇐ בנק DB Bank Account

הבנאדם שבא למשוך כסף הוא חיצוני למערכת ולכן הוא Actor = מצוייר מחוץ לscope של המערכת

לכל אחד מקבל Life line משלו.

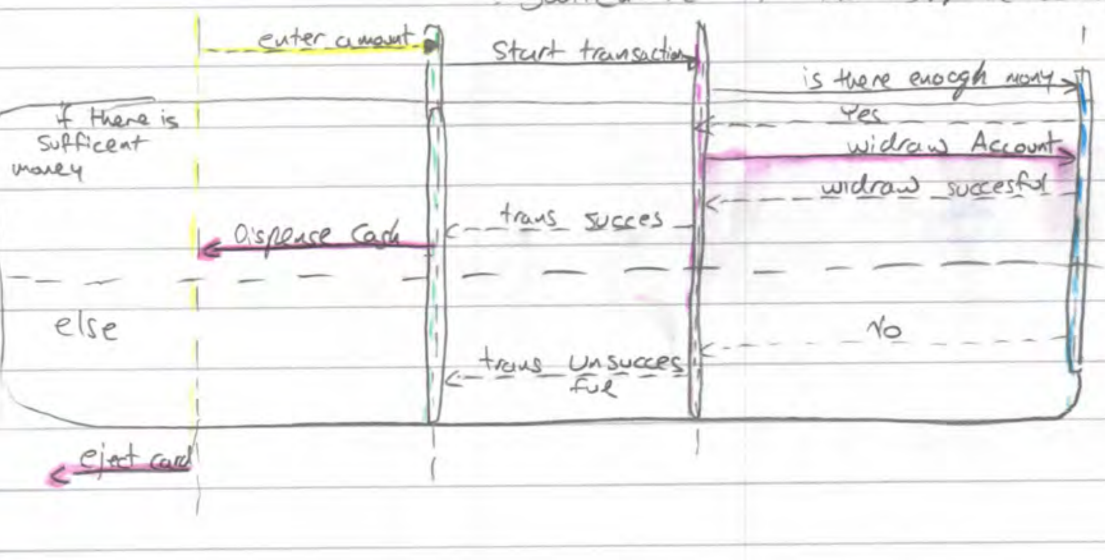


האינטראקציות מוצגות באמצעות message.

הם מראים את המידע שמועבר בין האובייקטים. עם כל הודעה אני יורד ב-life, אני מתקדם בזמן

כשאובייקט מעביר הודעה, כמו עונה על שאלה, זה מוצג ע"י -------> זוהי הודעה בין הצד המקבל לצד השולח

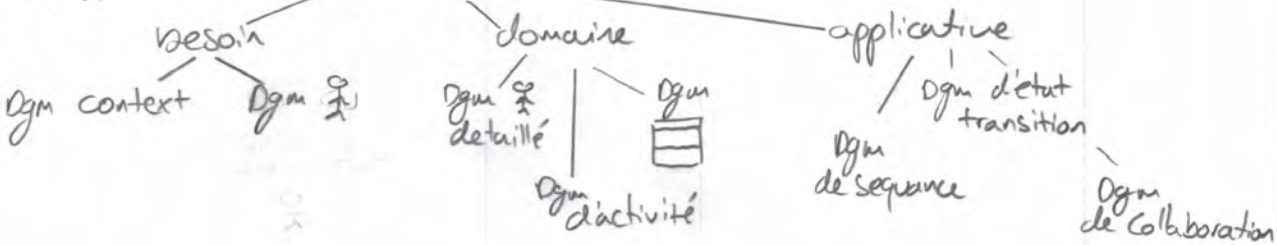
If: ברגע שאני רוצה להעביר את ה-notion של אם true... תעשה כך וכך ואם לא... תעשה כך וכך אני משתמש ב-Alternative frame. זה כמו if או Switch.



UML Unified Modeling Language

open

Dgm = Diagramme
CU = cas d'utilisation

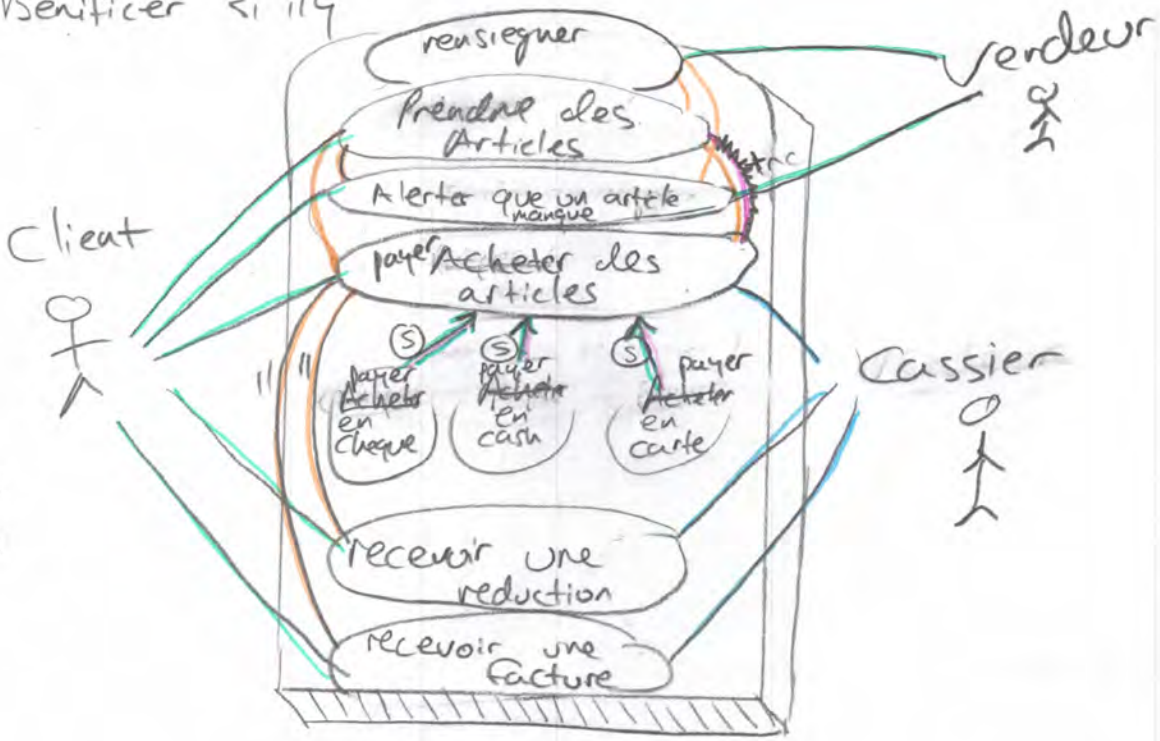


Logiciels:
StarUML
ArgoUML
BoUML
PowerDesigner

Dans un magasin processus de vente:

+ include
(S) spe
(//) extend

- client entre
- pass dans le rayon
- renseignement ou
- procedde a des essaye
- prende des articles si il y a dans le stock
- caisse tous les moyens accepté
- reduction Benificier si il y



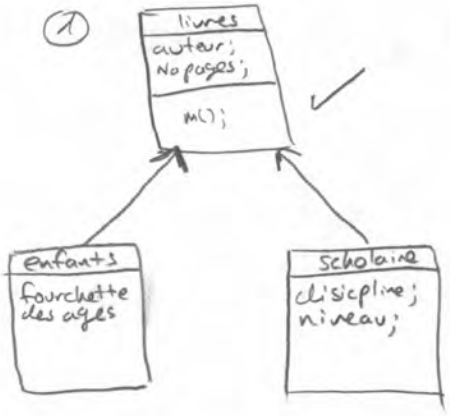
include
extend
speci

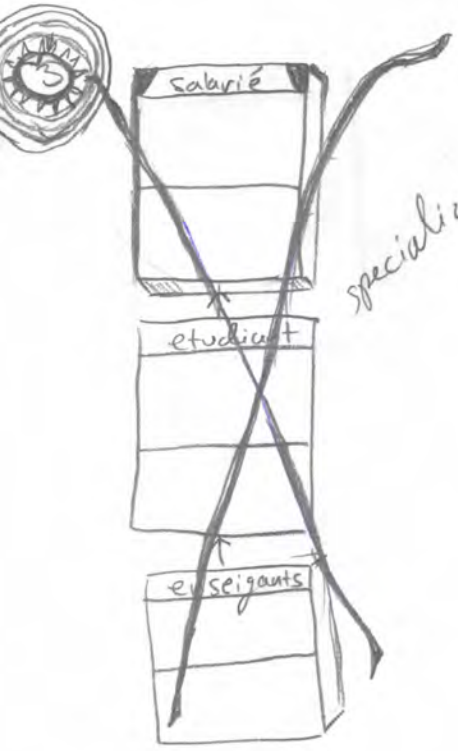
indique le type de relation: Agregation, Composition, Specialisation, simple

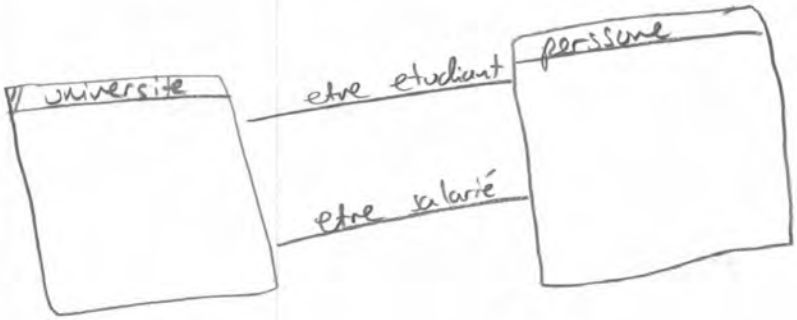
1) une ville peut eter une capitale. specialisation ✓
2) une transaction bours peut eter une achete /vente. specialisation ✓
3) une perssone utilise une langage pour coder. Agregation / simple relation
4) modem clavier ecran sont des peripherique dun pc. Composition

Answer Ex08.12.png

① qui : une librairie
service : vendre des livres — auteur
type de livres : enfants : fourchette des ages — nombre de pages.
scolaires : discipline, niveau





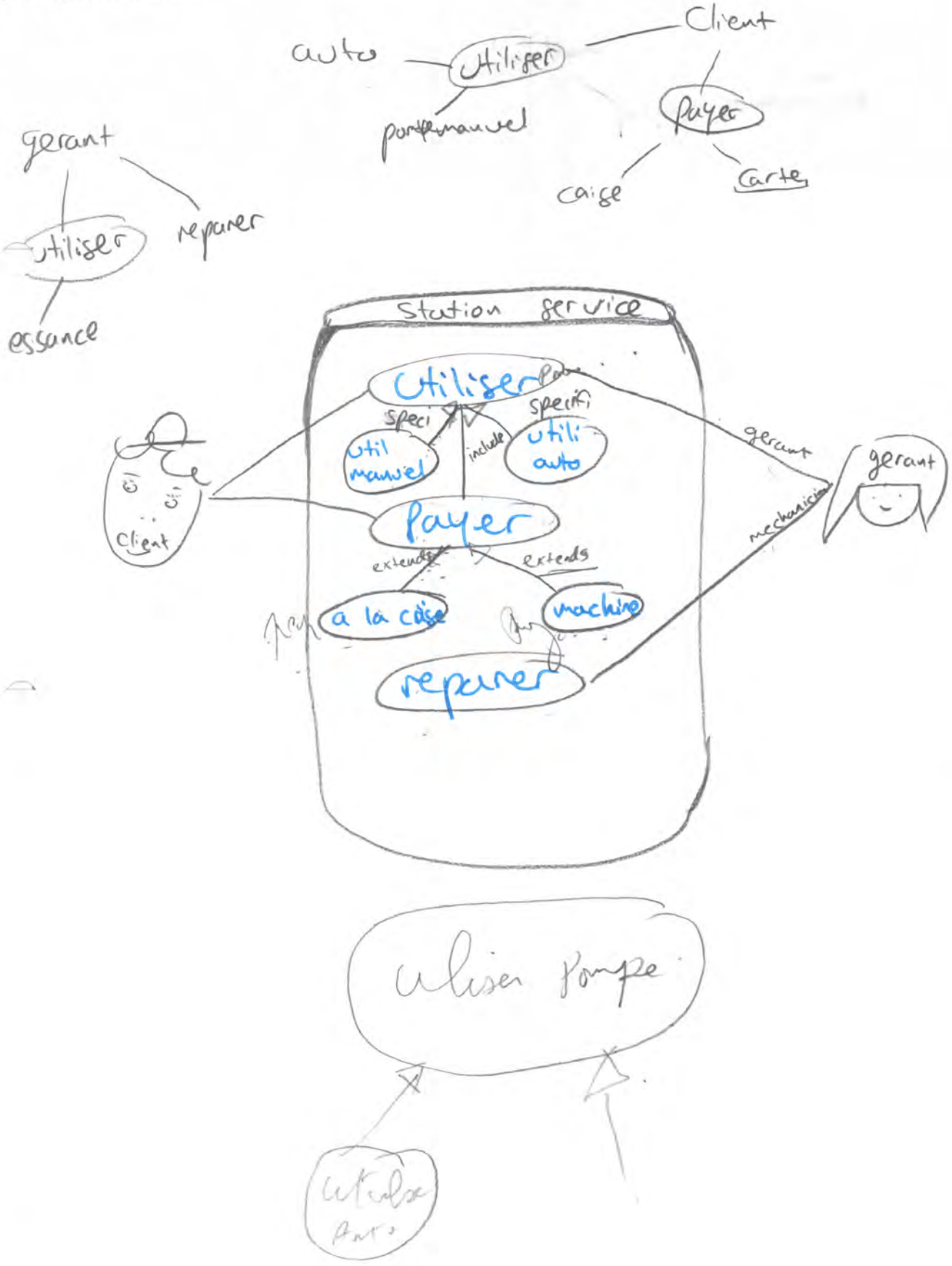


2h

#ex de diagram de sequence apres avoir lu le .ppt

System info → station service user case

le client peux utiliser le port manuel et payer a la caise de gerant ou utiliser des pompes automatique les gerant utilise le system info pour ses aperations: le gerant peuvent se servir lesance pour sa voiture. et dernierment la station a atelier d'entretien de veicle le gerant et aussi un mechanician.



SQL + JAVA = JDBC

Java DB Connectivity

1) import packages
2) Load and register your driver
3) connecting — getConnection
4) Writing a query (prepareStatment) (//set parameters if needed)
5) Exciting Query — excuteQuery -> exel
6) Processing result : 1. Process Output } the system does
2. Retrieve values }
7) closing statment — statments + resultSet
8) Clossing connection : — DB

services → Databases → New connection → New Driver → locate the driver location → fill: Host, port, username, password

Prepared Statment

create the Statment

PreparedStatment myst = myConn.prepareStatment (Here I put my query)

ex - 1) -11- ("select * from ... where")

Set parameter values -type -position -value

myst.setDouble (1, 8200)

type / position / value

exe

ResultSet myres = myst.executeQuery();

Update, Delete

JDBC : connect to a DB version I

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

public class sqls01 {

    public static void main(String[] args) {

        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        } catch (ClassNotFoundException ex) {
            System.err.println("Oops:ClassNotFound:" + ex.getMessage());
        }

        Connection connexion = null;
        try {
            String url = "jdbc:sqlserver://localhost:1433;"
                    + "databaseName=maBase;"
                    + "user=sa;password=sa";
            connexion = DriverManager.getConnection(url);
        } catch (SQLException ex) {
            System.err.println("Oops:Connexion:"+ ex.getMessage());
        }

        try {
            Statement stmt= connexion.createStatement();
            String query= "INSERT INTO maTable VALUES ( 5, 'EEE', 5.5);";
            int result= stmt.executeUpdate(query);
            System.out.println("Result:"+ result);
            stmt.close();
        } catch (SQLException ex) {
            System.err.println("Oops:SQL:"+
ex.getErrorCode()+"/"+ex.getMessage());
        }

        try {
            connexion.close();
        } catch (SQLException ex) {
            System.err.println("Oops:close:"+ ex.getMessage());
        }

        System.out.println("Done!");
    }

}
```

JDBC : connect to aDB version II

```
import com.microsoft.sqlserver.jdbc.SQLServerDataSource;
import com.microsoft.sqlserver.jdbc.SQLServerException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

public class sqls02 {

    public static void main(String[] args) {

        Connection connexion = null;

        try {
            SQLServerDataSource ds = new SQLServerDataSource();
            ds.setUser("sa");
            ds.setPassword("sa");
            ds.setServerName("localhost");
            ds.setPortNumber(1433);
            ds.setDatabaseName("maBase");
            connexion = ds.getConnection();
        } catch (SQLServerException ex) {
            System.err.println("Oops:SQLServer:"
                    + ex.getErrorCode() + "/" + ex.getMessage());
        }

        try {
            Statement stmt = connexion.createStatement(
                    ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_READ_ONLY
            );

            String query = "SELECT * FROM maTable;";

            ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                System.out.println(rs.getInt("ref"));
                System.out.println(rs.getString("text"));
                System.out.println(rs.getFloat("solde"));
                System.out.println("--");
            }
            System.out.println("-----");
            while (rs.previous()) {
                System.out.println(rs.getInt("ref"));
                System.out.println(rs.getString("text"));
                System.out.println(rs.getFloat("solde"));
                System.out.println("--");
            }

            rs.close();
            stmt.close();
```

```
        } catch (SQLException ex) {
            System.err.println("Oops:SQL:" + ex.getErrorCode() + "/" +
ex.getMessage());
        }

        try {
            connexion.close();
        } catch (SQLException ex) {
            System.err.println("Oops:close:" + ex.getMessage());
        }
        System.out.println("Done!");
    }

}
```

J2EE

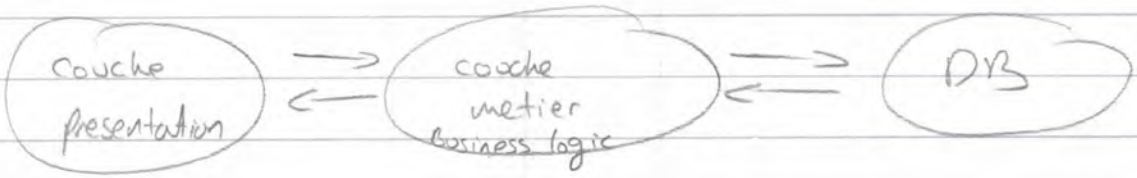
J'ai ?

type d'architectures :

- 1/3 : présentation, traitment, donné — stand-alone
- 2/3 : pour nombre reduit de Clients — tiers client, tiers Serveur (présentation / Données)
- n/3 : client — serveur web — server d'app — acces aux donné

couche traitment

présentation	metier
Jsp, servlet	EJB
JSF	



JSE

JME Mobile

JEE Entreprise

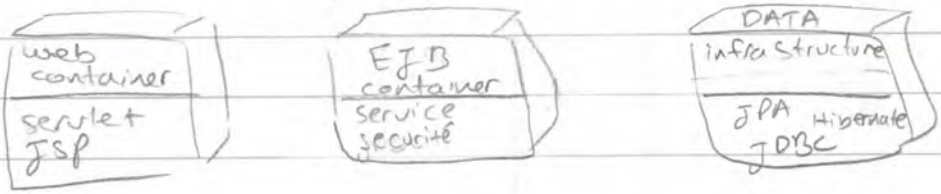
3 servers

HTTP → App server → DB

logique metiere → couche web

- on met jamais presentation dans la logique metiere

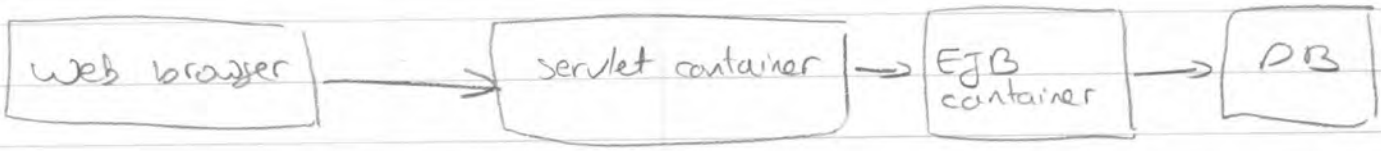
Servereur d'application



servlet : Java program that run on an HTTP web server

How ? ... extends HttpServlet ⇒ overide doGet()

Maven
outil de build Paradigme POM project Object Model
pom.xml : declaration des dépandance



ORM
object Relational Mapping
Beans can't be used in desktop app

JPA (entites)
entity = table in the DB
@ID Ele primere
@ Generate
@ Idclass
@ Column

@ quoi
@ modification
@ Lob
@ Basic
LAZY
EAGER

@one to many → List< >

Annotation

@Entity @ID @GeneratedValue

@Table @Column

The Class xxxx

@Entity

@Table (name="tablename") →

public class xxxx {

@Id

@Column (name="xxx_id") auto generated by DB

@GeneratedValue (strategy=GenerationType.IDENTITY)

private Integer id; → first column

private

@Column (name="name", length = 128, nullable = "false") varchar(128) can be null?

private String name; → Second col

private Floate price; →

create getters and setters

Run It

Create a class with a main method

final StandardServiceRegistry reg = new standardServiceRegistryBuilder()

.configure() // configure setting from the cfg.xml

.buid()

try {

① SessionFactory factory = new MetadataSource(reg) — a factory for session objects. 1 pos DB

.buildMetadata().buildSessionFactory();

② Session session = factory.openSession();

③ Transaction tran = session.beginTransaction();

④ Create object and set its attribute (object: entity)

⑤ session.save(object)

⑥ transaction.commit()

⑦ ssesion.close()

⑧ transact.close()

Generating Tables From JPA

- can be done by Hibernate
  where: cfg.xml or presistance.xml
  <property name="hbm2ddl.auto> create </property>
  When I run the app for the first time it will create. But after creating I need to change it to update

  It seems like I need to create a main method and inside
  EntityManagerFactory factory = Persistance.CreateEnt... (name of persistance)
  EntityManager em = factory.CreateEntityManager();
  UTUBE: Hibernate create tables from Entites (Forward Engineering)

- pom.xml
  For it to work I need the following dependencies
  ① hibernate-core
  ② mysql-connector-java

  in the main resources I will create a new xml file: hibernate.cfg.xml

  cfg.xml
  <hibernate-configuration>
  <session-factory>
- <property name="connection.url">jdbc:mysql://localhost
  <property name="connection.username">root </p
  < - password">pixi2417 </
  <property name="dialect">org.hibernate.dialect.MySQL5Dialect </
  <property name="hibernate.show_sql">true</ req -> pk 26 35
  <property name="hibernate.format_sql">true / SQL
  <property name="hibernate.hbm2ddl.auto">create </pr >
  <mapping class="packadgename.classname"> → of the entity

- update: if I want to add columns or add data, if I made changes Entity
- create-drop: use for testing. drop when the session ends / sessionFactory
  create: if the table exists it drop it and then create
  validate: validate the schema, makes no changes to the DB
  none: Hibernate doesnt touch the schema use it when the DB = stable and in production

Entity Manager → interface !!!   App Server

interface   מגדיר את ה-methods שהמימוש שלהן implement ב-session bean

מאפשר crud בלי SQL request. יש בו הרבה methods

add → presist()
delete → delete()
Flush() → Force la synchron avec le DB
UPDATE → merge()

} methods abstract — implement של מימוש ב-session bean

EJB

Injection de dependance : Design Pattern.   לכל סוגים

Dependance = service

@Persistance Context
EntityManger em;
↓ injection par interface

} in my Entity Manager I inject the service EntityManager em;

Session Bean

@Stateless : ↗ Peformance ↘ temp d'exe ~~cache~~ stocké — client G יקבל ejb מאותו הקשר

@stateful : pas perstitance ~~pas stocké en cache~~ — client 1 = instance 1 (תמיד).

@ singleton : 1 seul instance en memorie pour tous les clients

class pojo implement serializable

serializable — מאפשר ל-class לשמור את ה-etat של ה-Object

JNDI Java Naming and Directory

מאפשר לתת שם (unique) ייחודי לכל service שלו

WIKIPEDIA

List of Java keywords

In the Java programming language, a **Keyword** is any one of 52 reserved words[1] that have a predefined meaning in the language; because of this, programmers cannot use keywords as names for variables, methods, classes, or as any other identifier.[2] Of these 52 keywords, 49 are in use, 1 is in preview, and 2 are not in use. Due to their special functions in the language, most integrated development environments for Java use syntax highlighting to display keywords in a different colour for easy identification.

```
public void processData() {
   do {
      int data = getData();

      if (data < 0)
         performOperation1(data);
      else
         performOperation2(data);
   } while (hasMoreData());
}
```

A snippet of Java code with keywords highlighted in blue and bold font

Contents

List

`_`
: Added in Java 9, the underscore has become a keyword and cannot be used as a variable name anymore.[3]

`abstract`
: A method with no definition must be declared as abstract and the class containing it must be declared as abstract. Abstract classes cannot be instantiated. Abstract methods must be implemented in the sub classes. The abstract keyword cannot be used with variables or constructors. Note that an abstract class isn't required to have an abstract method at all.

`assert` **(added in J2SE 1.4)**[4]
: Assert describes a predicate (a true–false statement) placed in a Java program to indicate that the developer thinks that the predicate is always true at that place. If an assertion evaluates to false at run-time, an assertion failure results, which typically causes execution to abort. Optionally enable by ClassLoader method.

`boolean`
: Defines a boolean variable for the values "true" or "false" only. By default, the value of boolean primitive type is false. This keyword is also used to declare that a method returns a value of the primitive type `boolean`.

`break`
: Used to end the execution in the current loop body.

`byte`
: The `byte` keyword is used to declare a field that can hold an 8-bit signed two's complement integer.[5][6] This keyword is also used to declare that a method returns a value of the primitive type `byte`.[7][8]

`case`
: A statement in the `switch` block can be labeled with one or more `case` or `default` labels. The `switch` statement evaluates its expression, then executes all statements that follow the matching `case` label; see `switch`.[9][10]

`catch`
: Used in conjunction with a `try` block and an optional `finally` block. The statements in the `catch` block specify what to do if a specific type of exception is thrown by the `try` block.

`char`
: Defines a character variable capable of holding any character of the java source file's character set.

`class`
: A type that defines the implementation of a particular kind of object. A class definition defines instance and class fields, methods, and inner classes as well as specifying the interfaces the class implements and the immediate superclass of the class. If the superclass is not explicitly specified, the superclass is implicitly `Object (https://docs.oracle.com/javase/10/docs/api/java/lang/Object.html)`. The class keyword can also be used in the form Class.**class** to get a Class object without needing an instance of that class. For example, **String.class** can be used instead of doing **new String().getClass()**.

`const`
: Unused but reserved.

`continue`
: Used to resume program execution at the end of the current loop body. If followed by a label, `continue` resumes execution at the end of the enclosing labeled loop body.

`default`
: The `default` keyword can optionally be used in a switch statement to label a block of statements to be executed if no `case` matches the specified value; see *`switch`*.[9][10] Alternatively, the `default` keyword can also be used to declare default values in a Java annotation. From Java 8 onwards, the `default` keyword can be used to allow an interface to provide an implementation of a method.

`do`
: The `do` keyword is used in conjunction with `while` to create a do-while loop, which executes a block of statements associated with the loop and then tests a boolean expression associated with the `while`. If the expression evaluates to `true`, the block is executed again; this continues until the expression evaluates to `false`.[11][12]

`double`
: The `double` keyword is used to declare a variable that can hold a 64-bit double precision IEEE 754 floating-point number.[5][6] This keyword is also used to declare that a method returns a value of the primitive type `double`.[7][8]

`else`
: The `else` keyword is used in conjunction with `if` to create an if-else statement, which tests a boolean expression; if the expression evaluates to `true`, the block of statements associated

with the `if` are evaluated; if it evaluates to `false`, the block of statements associated with the `else` are evaluated.[13][14]

`enum` (added in J2SE 5.0)[4]
A Java keyword used to declare an enumerated type. Enumerations extend the base class `Enum (https://docs.oracle.com/javase/10/docs/api/java/lang/Enum.html)`.

`extends`
Used in a class declaration to specify the superclass; used in an interface declaration to specify one or more superinterfaces. Class X extends class Y to add functionality, either by adding fields or methods to class Y, or by overriding methods of class Y. An interface Z extends one or more interfaces by adding methods. Class X is said to be a subclass of class Y; Interface Z is said to be a subinterface of the interfaces it extends.
Also used to specify an upper bound on a type parameter in Generics.

`final`
Define an entity once that cannot be changed nor derived from later. More specifically: a final class cannot be subclassed, a final method cannot be overridden, and a final variable can occur at most once as a left-hand expression on an executed command. All methods in a final class are implicitly `final`.

`finally`
Used to define a block of statements for a block defined previously by the `try` keyword. The `finally` block is executed after execution exits the `try` block and any associated `catch` clauses regardless of whether an exception was thrown or caught, or execution left method in the middle of the `try` or `catch` blocks using the `return` keyword.

`float`
The `float` keyword is used to declare a variable that can hold a 32-bit single precision IEEE 754 floating-point number.[5][6] This keyword is also used to declare that a method returns a value of the primitive type `float`.[7][8]

`for`
The `for` keyword is used to create a for loop, which specifies a variable initialization, a boolean expression, and an incrementation. The variable initialization is performed first, and then the boolean expression is evaluated. If the expression evaluates to `true`, the block of statements associated with the loop are executed, and then the incrementation is performed. The boolean expression is then evaluated again; this continues until the expression evaluates to `false`.[15]

As of J2SE 5.0, the `for` keyword can also be used to create a so-called "enhanced for loop",[16] which specifies an array or `Iterable (https://docs.oracle.com/javase/10/docs/api/java/lang/Iterable.html)` object; each iteration of the loop executes the associated block of statements using a different element in the array or `Iterable`.[15]

`goto`
Unused

`if`
The `if` keyword is used to create an if statement, which tests a boolean expression; if the expression evaluates to `true`, the block of statements associated with the if statement is executed. This keyword can also be used to create an if-else statement; see *`else`*.[13][14]

`implements`
Included in a class declaration to specify one or more interfaces that are implemented by the current class. A class inherits the types and abstract methods declared by the interfaces.

`import`
Used at the beginning of a source file to specify classes or entire Java packages to be referred to later without including their package names in the reference. Since J2SE 5.0, `import` statements can import `static` members of a class.

`instanceof`
A binary operator that takes an object reference as its first operand and a class or interface as its second operand and produces a boolean result. The `instanceof` operator evaluates to true if and only if the runtime type of the object is assignment compatible with the class or interface.

`int`
The `int` keyword is used to declare a variable that can hold a 32-bit signed two's complement integer.[5][6] This keyword is also used to declare that a method returns a value of the primitive type `int`.[7][8]

`interface`
Used to declare a special type of class that only contains abstract or default methods, constant (`static final`) fields and `static` interfaces. It can later be implemented by classes that declare the interface with the `implements` keyword. As multiple inheritance is not allowed in Java, interfaces are used to circumvent it. An interface can be defined within another interface.

`long`
The `long` keyword is used to declare a variable that can hold a 64-bit signed two's complement integer.[5][6] This keyword is also used to declare that a method returns a value of the primitive type `long`.[7][8]

`native`
Used in method declarations to specify that the method is not implemented in the same Java source file, but rather in another language.[8]

`new`
Used to create an instance of a class or array object. Using keyword for this end is not completely necessary (as exemplified by Scala), though it serves two purposes: it enables the existence of different namespace for methods and class names, it defines statically and locally that a fresh object is indeed created, and of what runtime type it is (arguably introducing dependency into the code).

`non-sealed`
Used to declare that a class or interface which extends a sealed class can be extended by unknown classes.[17]

`package`
Java package is a group of similar classes and interfaces. Packages are declared with the `package` keyword.

`private`
The `private` keyword is used in the declaration of a method, field, or inner class; private members can only be accessed by other members of their own class.[18]

`protected`

The `protected` keyword is used in the declaration of a method, field, or inner class; protected members can only be accessed by members of their own class, that class's subclasses or classes from the same package.[18]

`public`
The `public` keyword is used in the declaration of a class, method, or field; public classes, methods, and fields can be accessed by the members of any class.[18]

`return`
Used to finish the execution of a method. It can be followed by a value required by the method definition that is returned to the caller.

`short`
The `short` keyword is used to declare a field that can hold a 16-bit signed two's complement integer.[5][6] This keyword is also used to declare that a method returns a value of the primitive type `short`.[7][8]

`static`
Used to declare a field, method, or inner class as a class field. Classes maintain one copy of class fields regardless of how many instances exist of that class. `static` also is used to define a method as a class method. Class methods are bound to the class instead of to a specific instance, and can only operate on class fields. (Classes and interfaces declared as `static` members of another class or interface are actually top-level classes and are *not* inner classes.)

`strictfp` (added in J2SE 1.2)[4]
A Java keyword used to restrict the precision and rounding of floating point calculations to ensure portability.[8]

`super`
Inheritance basically used to achieve dynamic binding or run-time polymorphism in java. Used to access members of a class inherited by the class in which it appears. Allows a subclass to access overridden methods and hidden members of its superclass. The `super` keyword is also used to forward a call from a constructor to a constructor in the superclass. Also used to specify a lower bound on a type parameter in Generics.

`switch`
The `switch` keyword is used in conjunction with `case` and `default` to create a switch statement, which evaluates a variable, matches its value to a specific `case`, and executes the block of statements associated with that `case`. If no `case` matches the value, the optional block labelled by `default` is executed, if included.[9][10]

`synchronized`
Used in the declaration of a method or code block to acquire the mutex lock for an object while the current thread executes the code.[8] For static methods, the object locked is the class's `Class`. Guarantees that at most one thread at a time operating on the same object executes that code. The mutex lock is automatically released when execution exits the synchronized code. Fields, classes and interfaces cannot be declared as *synchronized*.

`this`
Used to represent an instance of the class in which it appears. `this` can be used to access class members and as a reference to the current instance. The `this` keyword is also used to forward a call from one constructor in a class to another constructor in the same class.

`throw`
Causes the declared exception instance to be thrown. This causes execution to continue with the first enclosing exception handler declared by the `catch` keyword to handle an assignment compatible exception type. If no such exception handler is found in the current method, then the method returns and the process is repeated in the calling method. If no exception handler is found in any method call on the stack, then the exception is passed to the thread's uncaught exception handler.

`throws`
Used in method declarations to specify which exceptions are not handled within the method but rather passed to the next higher level of the program. All uncaught exceptions in a method that are not instances of `RuntimeException` must be declared using the `throws` keyword.

`transient`
Declares that an instance field is not part of the default serialized form of an object. When an object is serialized, only the values of its non-transient instance fields are included in the default serial representation. When an object is deserialized, transient fields are initialized only to their default value. If the default form is not used, e.g. when a *serialPersistentFields* table is declared in the class hierarchy, all `transient` keywords are ignored.[19][20]

`try`
Defines a block of statements that have exception handling. If an exception is thrown inside the `try` block, an optional `catch` block can handle declared exception types. Also, an optional `finally` block can be declared that will be executed when execution exits the `try` block and `catch` clauses, regardless of whether an exception is thrown or not. A `try` block must have at least one `catch` clause or a `finally` block.

`void`
The `void` keyword is used to declare that a method does not return any value.[7]

`volatile`
Used in field declarations to guarantee visibility of changes to variables across threads. Every read of a volatile variable will be read from main memory, and not from the CPU cache, and that every write to a volatile variable will be written to main memory, and not just to the CPU cache.[21] Methods, classes and interfaces thus cannot be declared *volatile*, nor can local variables or parameters.

`while`
The `while` keyword is used to create a while loop, which tests a boolean expression and executes the block of statements associated with the loop if the expression evaluates to `true`; this continues until the expression evaluates to `false`. This keyword can also be used to create a do-while loop; see *do*.[11][12]

Reserved Identifiers

The following identifiers are not keywords, however they are restricted in some contexts:

`permits`
The permits clause specifies the classes that are permitted to extend a sealed class.[22]

`record`

`sealed`
A sealed class or interface can only be extended or implemented only by classes and interfaces permitted to do so.[23]

`var`
: A special identifier that cannot be used as a type name (since Java 10).[24]

`yield`
: Used to set a value for a switch expression

Reserved words for literal values

`true`
: A boolean literal value.

`false`
: A boolean literal value.

`null`
: A reference literal value.

Unused

`const`
: Although reserved as a keyword in Java, `const` is not used and has no function.[2][25] For defining constants in Java, see the `final` keyword.

`goto`
: Although reserved as a keyword in Java, `goto` is not used and has no function.[2][25]

See also

- Java annotation

References

1. https://docs.oracle.com/en/java/javase/15/docs/specs/sealed-classes-jls.html#jls-3.9
2. "Java Language Specification - Section 3.9: Keywords" (https://docs.oracle.com/javase/specs/jls/se11/html/jls-3.html#jls-3.9). *The Java Language Specification*. Oracle. 2018-08-21. Retrieved 2018-12-25.
3. Goetz, Brian. "Warning about single underscore identifier" (http://openjdk.5641.n7.nabble.com/Warning-about-single-underscore-identifier-td145974.html#a146426). *OpenJDK Lambda Development*.
4. "Java Language Keywords" (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html). *The Java Tutorials*. Sun Microsystems, Inc. Retrieved 2017-07-24.
5. "Primitive Data Types" (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html). *The Java Tutorials*. Sun Microsystems, Inc. February 14, 2008. Retrieved 2008-12-03.
6. Flanagan 2005, p. 22.
7. "Returning a Value from a Method" (https://docs.oracle.com/javase/tutorial/java/javaOO/returnvalue.html). *The Java Tutorials*. Sun Microsystems, Inc. February 14, 2008. Retrieved 2008-12-03.
8. Flanagan 2005, pp. 66-67.
9. "The switch Statement" (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html). *The Java Tutorials*. Sun Microsystems, Inc. February 14, 2008. Retrieved 2014-12-18.