Ch10 P2

Big Integer , Big Decimal

ده لو عايز ياخد مساحة أكبر (قيم) من الـ int c decimal العاديين
كما immutable بردو لو عايز تعدل فيهم بيتعمل فيه object
عليهم pointer

→ Big Decimal c = a.multiply (b);
الجافا مش بتدعم الـ overloading في الضرب والطرح والقسمة و
هكذا

→ Factorial Example
141! = 131! x 14 من اول 141! والنواتج بتقل وده غلط لانها
ودى حاجة اسمها overflow بمعنى النواتج مش بتزيد عن
حجم الداتا نفسه يعنى لو int و زود عن حجمها بيحصل overflow
بالتالي بنستخدم Big Integer c Big Decimal .

→ BigInteger result = (BigInteger = new 1)
بدل ما يعمل كدة
BigInteger result = BigInteger.ONE;

→ (i + " ") بتحول الـ int لـ string

String Class P3

String message = "Welcome";
في الـ background فعليا ده اللي بيحصل
String message = new String ("Welcome");

String Functions

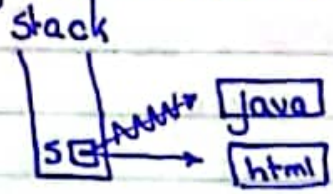
→ .length at( ) → عدد الحروف

→ .Char at (1) → لو ادتيه رقم يرجع الحرف اللي في المكان

→ (concat) → بتحط الجاحة جنب بعض

→ Substring (index) , substring (start , end) → تديله البداية والنهاية وهو يرجع كل الأرقام ماعدا الأخير

→ String are immutable / مهمة جداً جداً



والـ garbage collector
بتشيل كلمة java مع الوقت

→ Interned String

لو اثنين variables ~~بيشاوروا على نفس~~ شايلين نفس القيمة
الـ JVM بتخليهم الاثنين يشاوروا على نفس الحاجة وده لجودة الأداء والتحسين > بس تحذير <

```
String s1 = "Welcome";
String s2 = new String("Welcome");
String s3 = "Welcome";
```

لأن s2 كدة عملت new object

s1 == s2 (false)
s1 == s3 (true)

> نصيحة < منستخدمش (==) بتاعة الـ string عشان في بحالات وانت وحظك

كلمة ; ("") = new String لما نيجي نستخدمها String
حتى لو القيمة متكررة ، هيكون كل object بيشاور على مكان لوحده.

s1 == s2 false لان (==) بتقارن المكان

→ s1.equals(s2); كدة ده لو عايز تقارن القيم الي جوة
↳ true

→ int [] a = {1,2,3};
int [] b = {1,2,3};
System.out.println(a==b); // بتقارن نفس الأماكن ولا ايه
↳ output = False
System.out.println(a.equals(b));
↳ // هنا قارن مكان array a مكان b
output = false
System.out.println(Arrays.equals(a,b));
↳ // هنا قارن قيم الـ array ببعضها
output = true.

Summary

1 In Strings	s1.equals(s2)	يفضل استخدام
2 In Arrays	Arrays.equals(a,b)	يفضل استخدام

→ Replacing and Splitting Strings

* s1.replace("!","↓") بيبدل الحرف القديم بالجديد
(! قديم، ↓ جديد)
* s1.replaceFirst("!","\") بتبدل أول مكان بس
* s1.replaceAll("!","@") بتبدل كل الحروف
* s1.split بيجيبلك array فيها الحاجات والي بيفصل بينهم

Examples

	Outputs
* "Welcome".replace('e','A')	WAlcomA
* "Welcome".replaceFirst("e","AB")	WABlcome
* "Welcome".replace("e","AB")	WABlcomAB
* "Welcome".replace("el","AB")	WABcome

→ Splitting a String

```
String [] tokens = "Java#HTML#PerL".split("#", 0);
for (int i = 0; i<tokens.length; i++)
  System.out.print(tokens[i] + " ");
```

Output= Java HTML Perl

splite (" ↓ " , ↓)
الي هتفصل بيه Limit

Limit
- =0 هيفصل كله عن بعضه
- <0 هيفصل كام كلمة على عدد الرقم بس من اليمين
- >0 هيفصل كام كلمة على عدد الرقم بس من الشمال

P4

→ Regular Expressions < بتدور على pattern معينة >

* "Java".matches("Java"); // true
* "Java".equals("Java"); // true
* "Java is cool".matches("Java.*"); // true

. ← بعد الكلمة
لـ معناها
* ← صفر او اكثر من الحروف

< زمان > ↔ < دلوقتي >

عايزين نبدل كل $ + c # بـ "NNN"

دلوقتي هتاخدم كلهم بال regular expressions و تبدلهم علطول

↔

كنا هنعمل 3 replace كل مرة تديله العلامة والحرف.

```
String s = "a+b$#c".replaceAll("[$+#]","NNN");
String [] tokens = "Java,C?,C#,C++".split("[.,:,?]");
```

→ How to Convert Characters and Numbers to String

1. String.valueOf(5.44) — Output '5','.','4','4' string of characters
2. (5.44 + "") — Output "

Slide 54 # مهم جدا (كلها احفظها صم)

→ String s = "Java Java Java" .replaceAll ("v\\w", "wi") ;
بدل كل v و بعدها حرف واحد (\w) بـ wi

Output Jawi Jawi Jawi

→ String s = "Java Java Java" .replaceFirst ("v\\w", "wi") ;
هتبدل اول v\\w بس بـ wi

Output Jawi Java Java

→ String[] s = "Java1 HTML 2 Perl" . split ("#\\d") ;
لـ افصل عند اي رقم

→ String builder بيعالج فكرة ان الـ String immutable

StringBuilder (capacity:int)
by default capacity 16 هتاخد

* append → + بدل ما تعمل + (معنيش هنا) Slide 48

StringBuilder . append ("Java") ;
هيزود كلمة Java على الكلمة اللي عندك

* insert (index , String) ;
بتاخد المكان اللي هتزود فيه والحاجة اللي هتزودها

StringBuilder . insert (11 , "HTML and ") ;
هيروح عند مكان رقم 11 ويحط HTML and

* StringBuilder .delete (8,11) بيمسح بين ١١و٨
* StringBuilder .deleteCharAt (8) هيمسح الحرف الي في مكان ٨
* stringBuilder.reverse ( ) لو عايز تعكس كلمة
  Welcome ⟷ emocleW
* StringBuilder.replace (11,15,"HTML")
  المكان الى بين ١٥و١١ ابدله بكلمة HTML
* StringBuilder.SetCharAt (0,'W') هتبدل اول حرف بـ W
* toString ( ) : String تحول الكلمة لـ string
* capacity ( ) ترجع الـ capacity كانت قد ايه
* CharAt (index) الحرف الى فى index رجعهولي
* length ( ) الطول
*
*

Palindrome Ignore non Alphanumeric

بيقرا الكلمة ويقولك الي من ورا لقدام زي ما قدام لورا
ولا لا (boolean)

String s1 = filter(s); بيشيل اي حاجة مش alphanumeric

→ StringBuilder طالما هتكتب في الـ string استخدم

→ String للتعامل الثابت

Ch 11 P1

→ Encapsulation :- ربط الـ methods والـ variables في نفس الـ class

→ Abstraction :- زي الـ Functions و الـ private variables الي بتكون internal interface .

→ Inheritance :- لمنع التكرار (redundancy) & reusability

- java.util.date() بتحط التاريخ الحالي لـ variable

- superclass constructors inherited ? No , but invoked automatically implicitly → تناديه بنفسك
explicitly → automatic
لو الـ super class فيه أكثر من constructor هتنادي على الـ Constructor with no-argument

- public A() { } == public A() { super(); }
public A(double d) { } == public A(double d) { super(); }

→ Caution <تحذير> لازم تنادى على الـ super constructor
Syntax error لازم يكون اول حاجة (أول سطر) بدلما بيي

→ أول ما نعرف object من class وارث من مكان تاني لازم نعرف انه أول حاجة بيعملها بينادى على super constructor

→ <تحذير مهم> لو class وارث من class معندهوش غير واحد constructor وبياخد parameters هيعمل error هو لازم ينادى الأول على constructor فاضي لو مش موجود اعملك

→ toString( ) الأكثر استخداماً و بتستخدم في الطباعة الـ Strings

→ Override methods
بتلغي استخدام الـ method القديم و تستخدم الجديدة بنفس الـ syntax متغيرش اي تغيير بس الـ methods مش بتتغير في مكانها مش بتقدر استخدمها عادي لما تكون

→ instance method ↳ can be overridden only if accessible مش فانكشن كدي operator
→ static method ⟶ can be inherited
⟶ cannot be overridden

Overload لو اثنين method نفس الاسم بس parameter مختلف
Overrides اعمل methods مكانها باستخدام مختلف

→ Object class موجود اساسي في الـ java ومهم و كله بيورث منه ولو مكتبتهاش هي موجودة by default

→ public String toString ( ) {
return getClass( ). getName( ) + "@" +
Integer. toHexString(hashCode( ));

ده الي بيطلع لما تعمل فانكشن toString و مش يهوش حاجة و
override اكتر واحدة بيتعملها

→ Polymorphism < تعدد الأشكال > P2
↓ حاجة ليها نفس الاسم بس بتعمل أكتر من غرض

variable of supertype can refer to subtype object

→ لو عملت فانكشن بتاخد parameter (object) بالتالى يقدر ياخد
من اي حد في ال classes

→ O1 instance of Student
دى فانكشن بتشوف هى موجودة جوة ال class و لا ايه

→ object o = new Student( ) // implicit casting
Student o3 = Student o1 // explicit casting

→ int i = 5 ; float f = 3.2f ;
f = i ✓ // implicit casting
i = f x i = (int) f ; // explicit casting

هنا لو حطيت int جوة float عادى لأن float بيشيل مساحة
للأرقام الصحيحة بالتالى عادى العكس غلط يدي error

دلوقتي أنا عارف و متأكد ان student أكيد object (implicit)
بس معرفش ال O1 الي من نوع object هو student و لا ايه
فدى على مسؤوليتك

→ Compiler → matching method
→ JVM → in runtime error binds implementation of method

→ Generic programming برمجة عامة
→ Casting from Superclass to subclass
↳ implicit , explicits
→ Casting دايما يفضل استخدام الـ instance of قبل ما تنفذ عملية الـ
→ equals Methods class object موجودة جوة
* بتقارن الـ reference افرض انت مش عاجبك الطريقة دي ممكن تستخدم الـ equals التفضيلية الى override بتعمل

→ ArrayList class P3
* Linked List → dynamic size , arrayList
* < E > → generic type
ex. ArrayList < String > names = new ArrayList < >( );
String بقى arrayList الـ
ex. < String > = < > (√)
< String > = < String > (√)

→ arrayList name . contains ( )
بترجع true or false تشوف الحاجة اللي فى القوس فى الـ List ولا لأ

→ arrayList name . indexof (كلمة);
"الكلمة" في مكان رقم كام ؟

→ arrayList name . add (index , كلمة);

* index of ### الarrays بتبدي من [0] صفر
* الـ arrayList بتعمل shift لوحدها في الحاجة
* Listname . remove (كلمة)
بتشيل الكلمة من الـ list
* Listname . remove (1)
بتشيله الي في المكان رقم واحد
* Listname . toString ( ) ; بتطبع الـ list في [ ]
* array cityList [i] == arrayList citylist . get (i)
< لو حتطبع حبوة for Loop >
* List < > list1 = new list < > (Arrays . asList (array)) ;
حبوة الـ parameter بتاع الـ constructor
* String [ ] array1 = new String [list . size ( )] ;
list . toArray (array1) ;
↳ by reference مبدونة فأي تغيير
هيتم في النسخة الأصلية
* java . util . Collections . max بتجيب أكبر رقم في الـ arraylist
java . util . Collections . min بتجيب أصغر رقم في arraylist
* java . util . Collections . shuffle (list)
لو عايز ترتب الحاجة بطريقة عشوائية بدل ما تعمل 2 for Loop وتعمل swap هو عملها automatic

→ MyStack Class

* peek → بتشوف المكان الي عليه الدور مين بس مش بتشيله

* object o = peek( ); ≡ الاتنين بيساوا بعض
object o = list.get(getSize( ) - 1);

* الي يهمنا في جزء الـ data structure:
1. Generic Type ده مهم واستخدامه كتير
2. ArrayList and arrays الفرق بين

P4 → كنا خدنا في الـ access modifiers
private , public , default ~~, pro~~
هناخد واحد جديد اسمه protected

* protected → بيشوفوا الناس الي بتورث مني والي معايا في نفس الـ package

*
visibility increases →
private , none , protected , public
↳ if no modifier is used

private → انت بس تشوفه
default → انت + الناس الي معاك في package
protected → انت + الناس في package + الي بتورث منه
public → كله

→ Subclass Cannot Weaken Accessibility

لو معاك variable فوق ✊ ونادية تحت override علشان تعمل حنار ثم حنار انك تضعف accessibility وده عن طريق

visibility increases →
private , default , protected , public

1- لو ف public ممنوع تغيرها لأي حاجة قبلها في السهم (✓)
2- لو protected ممكن public بس (✓)
3- لو default ممكن protected و public بس (✓)
4- لو private ممكن default و protected و public (X)

- لو عملت عكس السهم تكون بتضعف الـ accessibility
- رقم 4 غلط جباحبا لأنها لو مش متشافة هتوصلها إزاي 🤔

→ final modifiers in oop

* الحجات الـ (3) final هتتعمل
1- Constant variable (PI)
2- method ← cannot be overridden by Subclasses (لو قد وارث مني ميزفش يغيرها)
3- class ← مش عايز حد يورث منه و يغير فيه بس تعمل منه objects عادي بس <كيفي كدة>
<من حقك تستخدمه بس متزودش عليه>

< NOTE >

static :- يكف بتاعة الـ class الحالي وممكن تغير قيمتها
final :- يكف بتاعة الـ class ممنوع تغير فيها أو تيجي جنبها

Ch 12 حزءين 1 Exceptions
2 Read, Write on text Files

Exceptions

* runtime error لو قسمت مثلا على صفر
ليها طريقتين
if condition
لو مش صفر طلع ناتج
return 1 لو صفر
Method
لو مش صفر اطبع
System.exit(0) لو صفر

quotient with Exception

* try (حبتي يا جافا الكود عرفتي تغير وبركة معرفتيش)
ادخلي على catch
catch (نفذ الحمله الي جوة)

* throw new ← ارمي حمله الـ exception الي بعدي
مين هيمسك الحمله ← الـ main
هتوديها try و catch والي هياخد الحمله الـ catch

* Input Mismatch Exception
بيظهر لو دخلت حاجه مش نفس نوع المطلوب (datatype)
منعيز try, catch البرنامج هيضرب

* JVM Exceptions نادراً حدوثها
↳ Error

* Unchecked Exceptions الجافا مش بتشترط علينا
نحطلهم try, catch

* Java Exceptions 1. Checked 2. Unchecked

* Unchecked :- Runtime exceptions + errors
(لا تشترط فيها الـ try catch)

* throws IOException — مش شرط ثنوي exception دى بتقول احتمال تكون هترمى exception

Syntax
throw new (Exception اسم / نوع) ( ) ;

* try , catch — ممكن يكون فيها أكثر من catch
بس الترتيب مهم جداً بتاع الـ catch
بس الأفضل اثناء تعمل catch للحاجة المتأكد من حدوثها الخاص (IOException) قبل العام (Exception)
هينفعش تحول الـ catch الـ Exception العامة بعد كدة الخاصة

> ترتيب الـ catch من الصغير للكبير <
من subclass لـ superclass

* try , catch — بروح على try مع اول error في الـ error الى فى الـ catch علطول
~~try مش بكمل باقي الـ try و لما اخلص اروح تاني الـ~~

* try {
statements ; }
catch (The Exception ex){
throw ex ; }

هى دى اعادة رمي الـ exceptions من جديد

* finally clause
مهم جداً ← بيتم تنفيذها دائماً سواء فيه exception او لا

```
try { statments ; }
catch (The Exception ex) {
    handling ex ; }
finally {
    finalStatements ; }
```

١ لو اشتغل الـ try :
- فيه error
  - يروح لـ catch
  - يروح لـ finally بعدها
- لو معفيش error
  - يروح ينفذ try ← وبعدها finally

- لو فيه throw تانية جوة اي catch فيوقع :-
  - هتروح لـ catch تنفذ اللي قبل الـ throw
  - بعدها هتروح لـ finally
  - بعدها ترجع تاني لـ throw

- لو فيه statement بعد الـ block finally :-
  - ~~لو فيه~~ لو فيه throw جوة catch
    - مش هتتنفذ الـ Statement
    - لو معفيش
      - هتتنفذ عادي

* Cautions when using Exceptions
برغم سهولة البرمجة بالـ exception إلا انك لازم تكون عارف إنها بتبطئ في runtime وبتاخد وقت كبير
Requires : → more time and resources
(roll back) . عشان بترجع تلف في الـ stack

* When to Throw Exception

لازم الـ throw تتعمل في method
لو عايز الـ user يحدد نوع الـ throw واختلاف الاحتمالات
اعمل exception obj و اعملك throw
لو هتقدر تعمل handle للـ error (exception)
جوة الـ method بيفضل تكملها و متكملش throw

* Custom Exception Classes

ممكن تعمل exception لبيك لو مفيش حاجة في الـ exceptions API شبالك معالك

→ File Class

Part 2

ده Class مش هتتعمل فيه read و Write
سلايد 53 ضرورية

* isDirectry( ) boolean
ده بيقول هو ده المسار ولا ايه

* isAbsolute ( ) جاي من اول الـ C C D مثلاً

* mkdir ( ) تعمل directory اكتر من آكشون

→ PrintWriter (بتكتب عن طريقه)

Scanner (بتقرأ عن طريقه)

سلايد 56

Java.io.PrintWriter ممكن بدل ما اقول اعمل
java.io.* ؛ فوق import اعمل java.io.Scanner

* output.close(); مهم جداً عشان تعمل FileUsave

* بيعمل حاجة اسمها try عشان لو نسيت الclose يعملها

Scanner Functions سلايد 58

* input.hasNext() لو فيه سطر فاضي
  - input.next() كانت بتقرأ String لحد أول space
  - input.nextLn() عشان تقرأ السطر كله

* Scanner استخدمناه بـ 3 طرق طول الترم
  - Scanner (System.in) بتاخد من الuser و بتقرأ
  - Scanner (File Path) بتاخد String فيه الFilepath
  - Scanner (URL.openStream()); بتاخد من صفحة على النت و بتبحث الـURL بتاعها

* Web Crawler ابحث URL وهو يعمل كل URL و يفتحه كامل Web Site

Crawler (url);
get.SubUrls( );
↳ بتاخد اي حاجة اولها https ،http