



المؤسسة العامة للتدريب التقني والمهني
Technical and Vocational Training Corporation

أساسيات برمجة الحاسب الآلي

م. عبدالمجيد العتيبي



الطبعة الأولى





الفهرس

0	تمهيد
٦	الوحدة الأولى :
١١ - ٧	مقدمة عن الحاسب - نظري
١٧ - ١٣	مقدمة عن الحاسب - عملي
١٨	الوحدة الثانية :
٢٣ - ١٩	أساسيات برمجة الحاسب - نظري
٢٨ - ٢٥	أساسيات برمجة الحاسب - عملي
٢٩	الوحدة الثالثة :
٣٥ - ٣٠	الخوارزميات وخرائط التدفق - نظري
٣٩ - ٣٧	الخوارزميات وخرائط التدفق - عملي
٤٠	الوحدة الرابعة :
٤٤ - ٤١	المتغيرات وإسناد القيم - نظري
٤٧ - ٤٦	المتغيرات وإسناد القيم - عملي
٤٨	الوحدة الخامسة :
٥٦ - ٤٩	الطباعة والادخال من خلال JOptionPane - نظري
٦٢ - ٥٨	الطباعة والادخال من خلال JOptionPane - عملي
٦٣	الوحدة السادسة :
٧٠ - ٦٤	التعبيرات المنطقية وكتابة جملة else , IF - نظري
٧٦ - ٧٢	التعبيرات المنطقية وكتابة جملة else , IF - عملي

**الوحدة السابعة :** ٧٧

كتابة جملة IF else , جملة Switch - نظري ٧٨ - ٨١

كتابة جملة IF else , جملة Switch - عملي ٨٣ - ٨٧

الوحدة الثامنة : ٨٨

كتابة جملة while , do while , جملة for - نظري ٨٩ - ٩٦

كتابة جملة while , do while , جملة for - عملي ٩٨ - ١٠١

الوحدة التاسعة : ١٠٢

الدوال في لغة الجافا - نظري ١٠٣ - ١٠٧

الدوال في لغة الجافا - عملي ١٠٩ - ١١٠

الوحدة العاشرة : ١١١

المصفوفات ذات البعد الواحد وذات البعدين - نظري ١١٢ - ١١٧

المصفوفات ذات البعد الواحد وذات البعدين - عملي ١١٩ - ١٢٤

الوحدة الحادي عشر : ١٢٥

الكيانات والكلاسات Classes and Objects - نظري ١٢٦ - ١٢٩

الكيانات والكلاسات Classes and Objects - عملي ١٣١ - ١٣٤

تمارين عامة ١٣٦ - ١٤٩

مصطلحات إنجليزية ١٥١ - ١٥٢

اختصارات الكمبيوتر ١٥٣

المراجع ١٥٤

عن المؤلف ١٥٥

النهاية ١٥٦

تمهيد

من المعلوم اليوم أن الحاسبات انتشرت انتشارا واسعا وكبيرا لدرجة أنها أصبحت في كل موقع وفي كل مكان ولا يمكن الاستغناء عنها بأي حال من الأحوال، وذلك لما تقوم به من أعمال كبيرة وعظيمة و لما تتمتع به من قدرة عالية على إجراء العمليات الحسابية وغيرها من العمليات في وقت قصير جدا، كما أنها تتميز بالقدرات العالية على معالجة الكم الهائل من البيانات حفظا وترتيباً واسترجاعا وبحثا وغيرها الكثير من العمليات. ونظرا لما سبق أصبح لزاما علينا - لكي نواكب هذا العصر ولكي ننهض بوطننا وشعبنا وأمتنا - أن نعرف الكثير عن هذه الحاسبات وكيف يمكن التعامل معها والاستفادة منها. ومن الوسائل التي تساعدنا على الاستفادة من هذه الحاسبات معرفة وإتقان إحدى لغات البرمجة المعروفة والمشهورة هذه الأيام، ومن هذه اللغات المشهورة والتي بدأت تستخدم على نطاق واسع لغة الجافا Java language وذلك لما تتمتع به من قدرة على العمل (التنفيذ) مع كل الحاسبات وسهولة كتابة البرامج المختلفة سواء منها البسيطة أو الكبيرة .

أهمية مهنة البرمجة

من المعلوم أن الذي يقوم بكتابة البرامج لحل المشكلات الكثيرة والمعقدة هم المبرمجون ولا يمكن الاستغناء عنهم بحال من الأحوال لأن دورهم مهم وحيوي وتكثر الحاجة لهم في شتى المجالات وذلك لعمل الآتي :-

- ١ - كتابة برامج وبناء الأنظمة المختلفة لحل المشاكل وتبسيط التعامل مع الحاسب .
- ٢ - المسؤولية الكاملة عن إصلاح ما يحدث من أعطال أو حل مشاكل الأنظمة المختلفة .
- ٣ - بناء واجهة المستخدم في كثير من اللغات والتطبيقات .
- ٤ - بناء أنظمة التشغيل مثل Windows , Unix وغيرها من النظم فمثلا تستخدم لغة C في بناء نظام التشغيل Unix .

الوحدة الأولى

مقدمة عن الحاسب

نظري

برنامج الحاسب

البرنامج هو عبارة عن مجموعة من التعليمات تعطى للحاسب للقيام بعمل ما مثل حساب مجموع قيم مختلفة، حساب المتوسط الحسابي، حساب مضروب عدد معينالخ والبرنامج هو الذي يحدد للحاسب كيفية التعامل مع البيانات للحصول على النتائج المطلوبة. والبرنامج يكتب بواسطة المبرمج (Computer Programmer) الذي يفهم المشكلة ويقترح الحل وينفذه لحل هذه المشكلة ويجب أن يكون البرنامج في مجموعه صحيحا وواضحا وليس فيه لبس أو غموض. والبرمجيات (Software) هي التي تسهل للمستخدم استخدام المكونات المادية (Hardware) بكفاءة وراحة ويمكن تقسيم البرمجيات إلى ثلاثة أنواع رئيسية وهي :

١- برامج التشغيل Operating System

وهي عبارة عن برامج تقوم بدور الوسيط بين المستخدم والمكونات المادية وهي تمكن المستخدم من استخدام المكونات المادية للحاسب بكفاءة وبراعة، كما أنها تساعد المستخدم في إنشاء نظام الملفات وغيرها .

أمثلة / النوافذ windows , VMS , Linux , Dos , Unix وغيرها .



٢- برامج التطبيقات Application Programs

وهي برامج تساعد في إنشاء كثير من التطبيقات مثل إنشاء قاعدة بيانات والرسم باستخدام الحاسب و غيرها .

أمثلة / برنامج الأوتوكاد – AutoCAD , الاكسيل – Excel , الأكسس – Access , الأوراكل Oracle -الفوتوشوب – Photoshop وغيرها .



٣- لغات البرمجة Programming Languages

هي التي تستخدم في بناء البرامج المختلفة وهي تتراوح من اللغات التي تتعامل مباشرة مع المكونات المادية للحاسب والأخرى التي تتطلب تحويلها من صورتها التي تكتب بها إلى صورة أخرى يستطيع الحاسب التعامل معها .

ويوجد العديد من لغات البرمجة المستخدمة اليوم وهذه اللغات يمكن تقسيمها إلى ثلاث أنواع رئيسية هي :

١- لغة الآلة Machine languages

٢- لغات التجميع Assembly languages

٣- لغات المستوى العالي High level languages

١- لغة الآلة Machine languages

وهي اللغة الوحيدة التي يفهمها الحاسب ويستطيع التعامل معها. وهذه اللغة تعتبر لغة خاصة لكل حاسب وقد تختلف من حاسب إلى آخر وهي تعتمد على المكونات المادية للحاسب نفسه، ولغة الآلة تتكون من مجموعة أرقام من بين ٠، ١ التي تعطي تعليمات للحاسب للقيام بمعظم العمليات الأساسية واحدة بعد الأخرى، وهي تختلف من حاسب إلى حاسب آخر ولذلك فإننا نجد أن نفس البرنامج الذي يعمل على حاسب معين قد لا يعمل على حاسب آخر يختلف عنه في المكونات المادية. ولغة الآلة من اللغات الصعبة في التعلم للإنسان حتى بالنسبة للمبرمجين لأنها عبارة عن مجموعة من الأرقام (٠، ١) فقط. وللتغلب على هذه الصعوبة تم اقتراح لغة أخرى تعتمد على استخدام اختصارات معبرة من اللغة الإنجليزية للتعبير عن العمليات الأولية التي يقوم بها الحاسب وهذه اللغة هي لغة التجميع.

مثال على لغة الآلة :

١	٠	١	٠	٠	١	١	١	٠	٠
---	---	---	---	---	---	---	---	---	---

٢- لغات التجميع Assembly languages

و هي لغة تستخدم اختصارات معبرة من اللغة الإنجليزية لتعبير بها عن العمليات الأولية التي يقوم بها الحاسب .

ونظرا لأن هذه اللغة تستخدم كلمات مختصرة من اللغة الإنجليزية فإنها تحتاج محولا لكي يحولها إلى لغة الآلة وهو ما يسمى المجمع assembler الذي يقوم بتحويل لغة التجميع إلى لغة الآلة كي يفهمها الحاسب ويستطيع تنفيذها، وبالرغم من تقليل المجهود الملقى على عاتق المبرمج للقيام بعملية البرمجة إلا أنه ما زالت توجد مشقة عند حل أبسط المسائل لأن ذلك يتطلب معرفة وكتابة العديد من التعليمات، وهذا ما دفع المبرمجين للتفكير في

لغات أخرى تقلل المجهود الكبير اللازم لكتابة الكثير من التعليمات فكانت لغات البرمجة ذات المستوى العالي .

مثال على هذه اللغة :

Load	A
Add	B
Store	C

٣- لغات المستوى العالي High level languages

وهذه اللغات كتبت بحيث تستخدم بعض الكلمات الإنجليزية العادية بنفس معانيها حيث يقوم كل أمر منها بتنفيذ العديد من الواجبات, وهذه اللغات كسابقتها تحتاج إلى مترجمات Compilers التي تقوم بتحويل التعليمات (الأوامر) إلى لغة الآلة, وهذه اللغات تستخدم العلاقات والعوامل الرياضية المتعارف عليها.

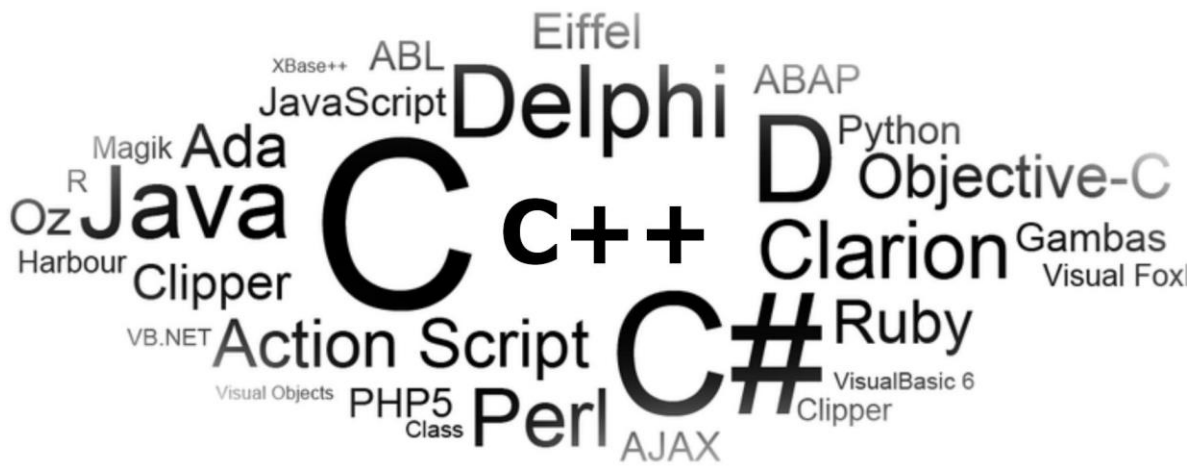
$$\text{Sum} = A + B + C$$



وهذه اللغات تعتبر سهلة ومرغوبة من وجهة نظر المبرمجين بالمقارنة بلغات التجميع ولغة الآلة وذلك لسهولة كتابتها وفهمها وحل المشاكل باستخدامها .

❖ ومن أمثلة لغات المستوى العالي :

لغة السي C	لغة الباسكال Pascal
لغة الجافا Java	لغة الفيجوال بيسك visual basic
لغة البي اتش بي PHP	لغة السي شارب C#



ومن المعلوم أن عملية تحويل البرنامج من لغة ذات مستوى عالي إلى لغة الآلة تستهلك وقتا ولذلك تم تطوير نسخ من لغات المستوى العالي بحيث تستخدم برنامج مفسر Interpreter والذي يقوم بترجمة الكود سطرا سطرا أثناء التنفيذ .

وبالرغم من أن البرامج المترجمة الناتجة من عملية الترجمة باستخدام المترجم compiler تكون أسرع في التنفيذ عن البرامج التي تستخدم المفسر (Interpreter) إلا أنه يفضل وجود نسخة من اللغة تعمل باستخدام المفسر وذلك لسهولة التغيير والحذف والإضافة والتصحيح. وبعد الانتهاء من كل التعديلات والوصول إلى نسخة نهائية فإنه يتم استخدام المترجم الترجمة البرنامج وإنتاج نسخة تنفيذية حتى تكون أسرع في التنفيذ بعد ذلك عند تشغيلها على الحاسب .

الوحدة الأولى

مقدمة عن الحاسب

عملي

ماهو برنامج Eclipse ؟

برنامج eclipse هو عبارة عن بيئة محورية متكاملة من اجل تطوير وتحديث البرمجيات المكتوبة بلغة "الجافا" ويعتبر البرنامج مجاني .

ويقع تحت ترخيص مفتوح المصدر مثله مثل الاندرويد , كما يدعم eclipse ايضا لغات برمجة اخرى مثل :- باسكال , البايثون بجانب دعمه للغة الجافا .

بالرغم من ان eclipse مجاني ومفتوح المصدر , الا انه ملئ بالوثائق التي تمكن المبرمج وتساعد على تطوير البرمجيات وحل المشاكل والصعوبات التي قد يواجهها المبرمج .

كما ان eclipse يمكنه العمل بنفس درجة الكفاءة على مختلف الانظمة والأجهزة .



تاريخ برنامج Eclipse ؟

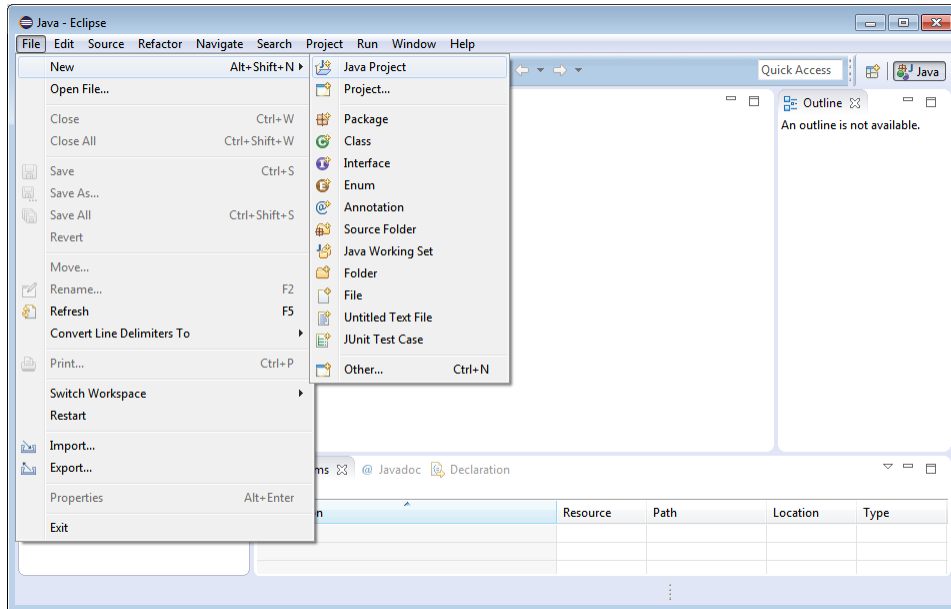
تم تطويره في العام ١٩٩٦ م , من قبل شركة IBM , وكان الغرض منه هو تحديث وتطوير التطبيقات والبرامج الخاصة بها المكتوبة فقط بلغة الجافا .

ثم بدءاً من العام ٢٠٠٤م , تم إتاحة eclipse كبرنامج مجاني مفتوح المصدر مما اتاح للجميع بتحديثه وتطويره وعمل التعديلات والتغييرات عليه بدون قيود او قوانين مانعة وراعية.

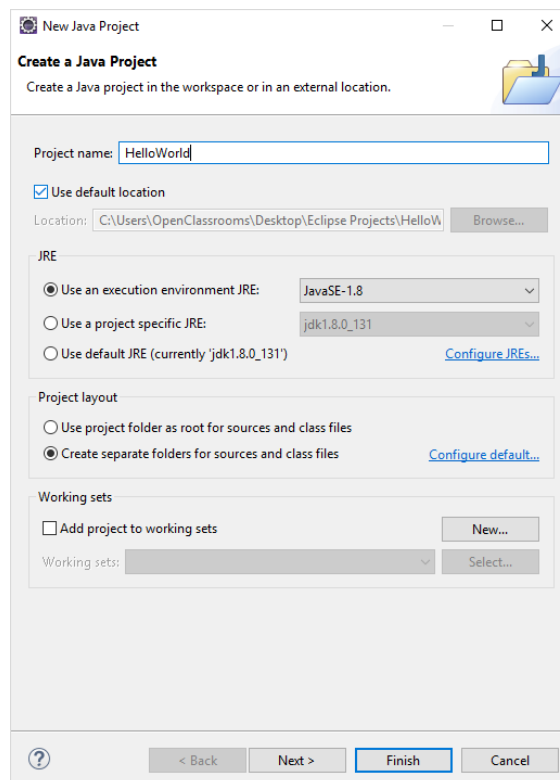
فيتعامل eclipse أيضاً مع الاضافات PLUGIN بشكل كبير وواسع حيث يمكنك كمبرمج او كمستخدم برمجة اضافاتك ومزجها في بيئة eclipse ثم يمكنك البدء في برمجة تطبيقاتك وبرامجك الخاصة بك .

طريقة إنشاء أول مشروع Project ؟

١. من قائمة **File** نختار **New** ومن ثم نختار **Java Project**

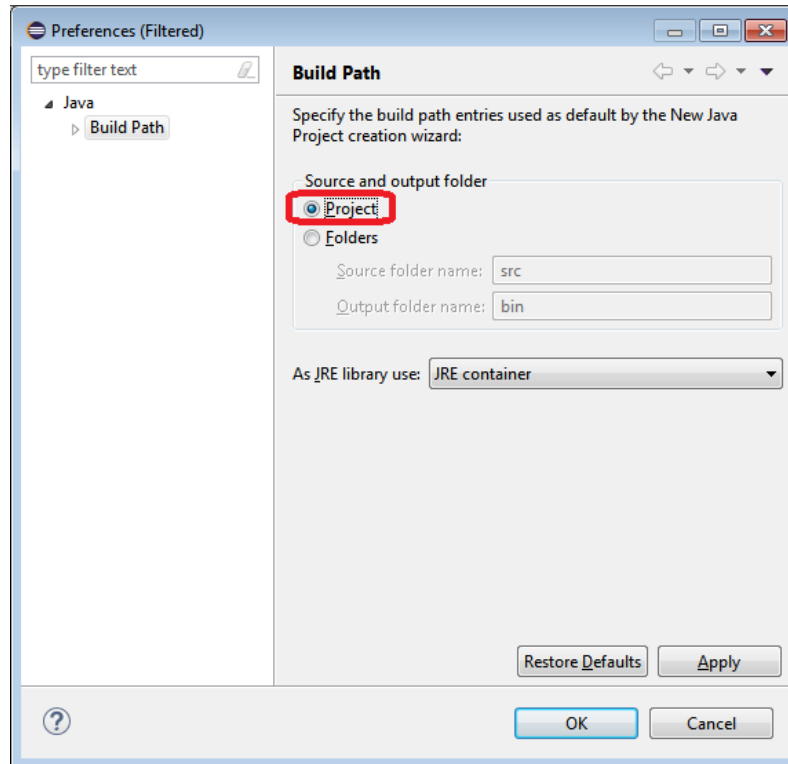


٢. سيتم عرض النافذة التالية , أدخل **اسم المشروع** وتحديد باقي الخيارات كما هو موضح في الصورة

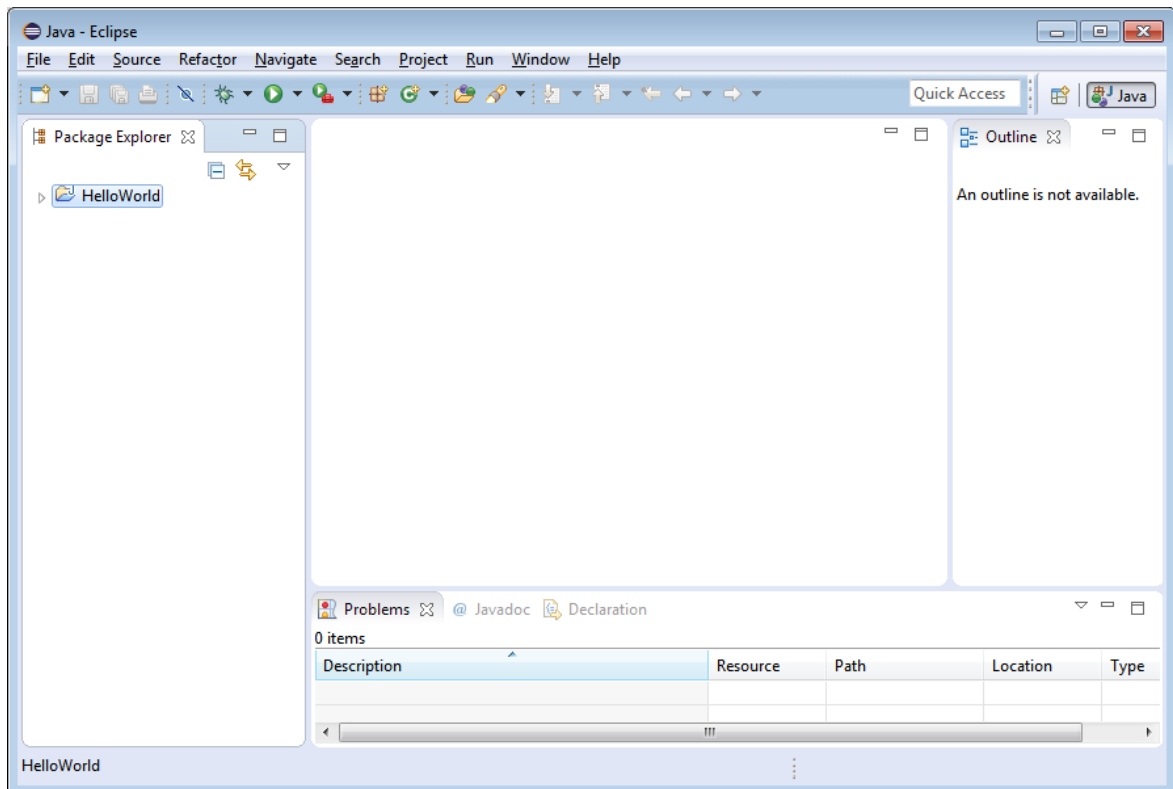




٣. بعد ذلك يتم إختيار **Project** ومن ثم الضغط على **OK**



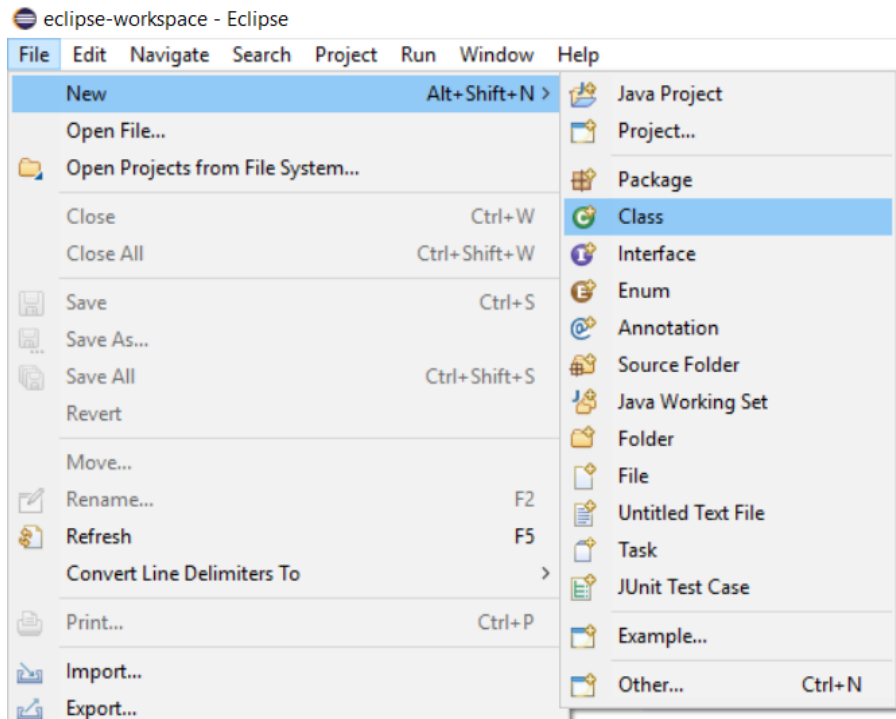
٤. مبروك , لقد قمت بإنشاء أول مشروع جافا الخاص بك !



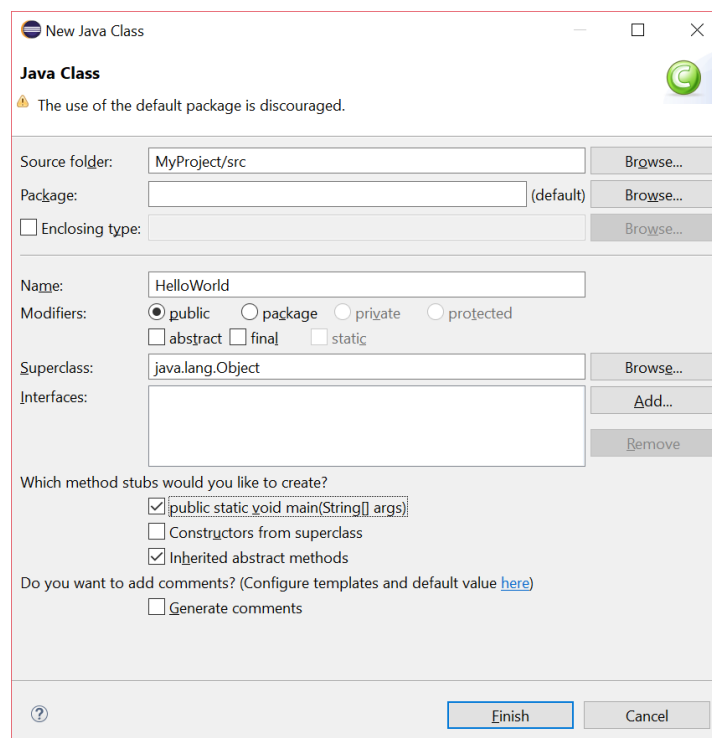


طريقة إنشاء كلاس Class ؟

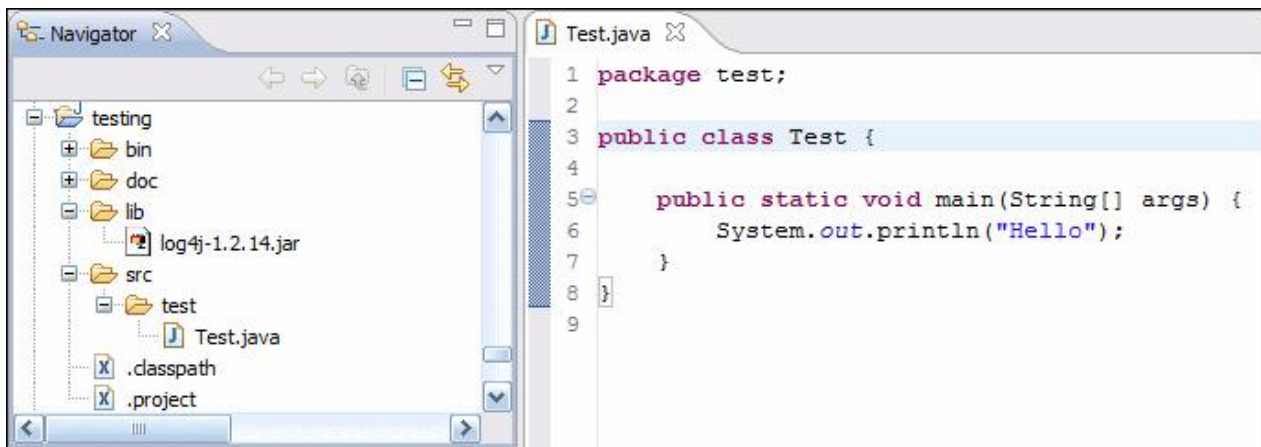
٥. من قائمة **File** نختار **New** ومن ثم نختار **Class**



٦. سيتم عرض النافذة التالية , أدخل اسم الكلاس وتحديد باقي الخيارات كما هو موضح في الصورة



٧. لقد أنشأنا Class جديد ونستطيع الآن كتابة الأوامر البرمجية



Package Test ;

هنا إسم المجلد او الملف الذي يحتوي على البرنامج

public class Test {

هنا اسم الكلاس (تعريف الفصيلة العامة او الكلاس)

public static void main (String[] args) {

في هذه المنطقة تكون الدالة الرئيسية

System.out.println ("Hello") ;

في هذه المنطقة يتم كتابة الأكواد الخاصة بنا

```
}
}
}
```

أقواس إغلاق الكلاس + الدالة
الرئيسية

الوحدة الثانية

أساسيات برمجة الحاسب

نظري

ماهي لغة جافا ؟

هي لغة برمجة عالية المستوى إبتكرها James Gosling في عام 1991 أثناء عمله في مختبرات شركة Sun Microsystems و ذلك لاستخدامها بمثابة العقل المفكر المستخدم لتشغيل الأجهزة الذكية.

عام 1995 تم تطويرها لبناء تطبيقات للويب، السيرفرات، سطح المكتب، الهواتف و الروبوتات. و هي تعمل على جميع و أهم أنظمة التشغيل مثل Windows ,MacOS ,Linux ,Unix ,Android إلخ .. و تعتبر من أشهر لغات البرمجة على الإطلاق .

حالياً ملايين الأجهزة الإلكترونية مبرمجة على لغة جافا و هذا يمثل شعار لغة جافا



مميزات لغة جافا بالنسبة للمطورين ؟

١. لها شعبية هائلة و هناك الكثير من المراجع لمن يريد تعلمها.
٢. بسيطة و تعلمها سهل مقارنةً مع غيرها من اللغات.
٣. شبيهة باللغتين C و C++ و لكن التعامل معها أسهل و سلس أكثر.
٤. إذا أنشأت برنامجاً باستخدام لغة جافا فإن البرنامج يعمل على أي نظام في العالم و هذا من أهم ما يدفعك لتعلمها .
٥. لغة جافا مطلوبة جداً في الخارج، إذا كنت تريد دخول سوق العمل فلغة جافا توفر لك الكثير من الفرص.

مميزات لغة جافا عن باقي لغات البرمجة ؟

١. **مادياً** : لن تدفع أي مبلغ لتعمل على لغة جافا, فهي مصدر مفتوح و مجانية و ستبقى مجانية مدى الحياة .

٢. **تقنياً** : تستطيع البرمجة بها حتى و لو كان حاسوبك ضعيفاً أو قديماً .

٣. **البساطة** : تعلمها سهل جداً بعد أن تفهم برمجة الكائنات .

٤. **العمل بحيادية** : يمكن لمترجم لغة جافا تقسيم أوامر البرنامج ليتنفذ على عدة معالجات بشكل متناسق باستخدام نظام JRE .

٥. **تعدد المهام** : جافا توفر لك تقنية الـ Multithreading و التي تسمح لك بجعل برنامجك قادراً على تنفيذ عدة أوامر مع بعض و بنفس الوقت .

٦. **سرعة الترجمة** : تتم ترجمة لغة جافا إلى أوامر يفهمها الجهاز بشكل جداً سريع و بدون أن يخرنها و يحاول تحليلها لأن الكائنات فيها تربط مع بعضها بشكل تدريجي مما يجعل الأوامر فيها تنفذ بسرعة .

٧. **معالجة الأخطاء** : جافا تعالج الأخطاء بطريقتين, أثناء كتابتك للكود تحاول أن تصلح لك أي أخطاء في كتابة الكود, و أثناء تشغيل الكود تخبرك بأي خطأ منطقي في حال حدوثه و الذي عليك معالجته بنفسك .

٨. **ديناميكياً** : صممت جافا لتكون أكثر ديناميكية من C و C++. ويمكن لبرامج جافا تحمّل كمية كبيرة من المعلومات وقت التشغيل و التي يمكن استخدامها للتحقق من الكود والتأكد إذا كان المترجم يستطيع أن يصل إلى الكائنات وقت التشغيل .



أسلوب ومبادئ كتابة الكود في الجافا ؟

١. حساسية حالة الأحرف بالإنجليزية Case sensitivity :

تعني أن لغة البرمجة تميز بين الأحرف الكبيرة و الأحرف الصغيرة .
مثال : Note و note ليسوا شيئاً واحداً .

٢. الفاصلة المنقوطة ; (Semicolon) :

حيث تستعمل هذه الفاصلات المنقوطة عند نهاية كل أمر في برامج جافا .
لابد وأن تنتهي كل جملة (سطر أو أمر) بعلامة (;)

قاعدة تسمية الاسماء في الجافا ؟

- i. أن يبدأ الاسم بحرف .
- ii. أن لا يبدأ برقم .
- iii. لا يحتوي على مسافة فارغة .
- iv. لا يكون من الأسماء المحجوزة (راجع قائمة الأسماء المحجوزة بالشكل ١) .
- v. يفضل أن يكون اسماً معبراً عن ما يقوم به الكائن .
- vi. لا يحتوي على أي حروف أو علامات خاصة أخرى غير المذكورة سابقاً .

ماهي التعليقات في الجافا Comments ؟

نستخدم التعليقات لنضع ملاحظات حول الكود الذي كتبناه فقط، لكي لا ننسى كيف برمجتنا الكود في حال أردنا مراجعته أو التعديل عليه بعد وقت طويل، كما أن التعليقات لا تؤثر إطلاقاً على الكود المكتوب .

تذكر أنك لست مجبراً على وضع تعليقات في برامجك، و لكننا ننصحك بوضع تعليقات دائماً حتى تساعدك في فهم الكود الذي كتبتة .

أنواع التعليقات في الجافا ؟

النوع الأول : التعليق بسطر واحد

إن هذا النوع من التعليق يتم بوضع علامتي (//) قبل السطر المراد تعليقه .

مثال

// هذا تعليق يتألف من سطر واحد

النوع الثاني : التعليق بعدة أسطر

يتم وضع الأسطر المراد شرحها أو التعليق عليها بين علامتيين (/*) و (*/) .

و هذا يعني ان هذه الأسطر هي عبارة عن تعليق , و لن يتم تنفيذها في البرنامج و لكنها وضعت للتوضيح , ان رؤية المترجم لـ (/*) تجعله يتجاهل كل ما يقابله حتى يصل لعلامة (*/) و يقوم بتنفيذ ما يليها .

مثال

/*
هذا تعليق
يتألف من
عدة أسطر
*/

الكلمات المحبوزة في لغة جافا ؟

public	final	abstract	Enum
return	finally	assert	private
short	float	boolean	while
static	for	boolean	else
strictfp	goto	byte	package
super	if	case	volatile
switch	implements	catch	double
synchronized	import	char	new
this	instanceof	class	void
throw	int	const	do
throws	interface	continue	native
transient	long	default	try

لائحة Escape Sequences الموجودة في لغة الجافا ؟

تعريفها	Escape Sequences
تضيف عدة مسافات في مكان وضعها	\t
تزيل الحرف الموجود قبلها	\b
تجعل المحتوى الذي يأتي بعدها ينزل سطر جديد	\n
تجعل الكود يبدأ في التنفيذ من عندها	\r
تضع فاصل بين المحتوى (أي تقسم المحتوى)	\f
لإضافة الرمز (') في مكان وضعها	\'
لإضافة الرمز (") في مكان وضعها	\"
تستخدم لإضافة أي حرف أو رمز من خلال الـ unicode	\u

الوحدة الثانية

أساسيات برمجة الحاسب

عملي



إنشاء أول برنامج في جافا :

سنقوم الآن بإنشاء برنامج بسيط مهمته فقط طباعة الجملة Welcome to java world .

مثال

```
public class Main {  
  
    public static void main(String[] args) {  
  
        // هنا قمنا بعرض الجملة  
        System.out.println("Welcome to java world") ;  
  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

Welcome to java world

- في المثال السابق قمنا بإستخدام الأمر **System.out.println()** لطباعة الجملة التي نريد عرضها .

مبادئ العرض الأساسية :

- عليك مراعاة المبادئ التالية عند استخدام دوال الطباعة :

- ✓ لعرض رقم، ضعه كما هو في دالة الطباعة .
- ✓ لعرض قيمة متغير، ضعه كما هو في دالة الطباعة .
- ✓ لعرض حرف أو كلمة أو نص، يجب وضعه بين " "

• مفهوم ال Concatenation ؟

Concatenation تعني سلسلة باللغة العربية. برمجياً تعني دمج عدة أشياء مع بعضها سواء كانت نصوص أو أرقام وجعلها تبدو شيئاً واحداً .
هنا قمنا بإنشاء برنامج يحتوي على أمر طباعة واحد , في هذا الأمر قمنا بدمج ثلاث كلمات و رقم, و عرضناهم مع بعضها كجملة واحدة .

مثال

```
public class Main {
    public static void main(String[] args) {
        // هنا قمنا بدمج ثلاث كلمات و رقم، و عرضناهم مع بعضهم كجملة واحدة
        System.out.println("Welcome " + "to " + "java " + 101);
    }
}
```

سنحصل على النتيجة التالية عند التشغيل

Welcome to java 101

• دوال العرض :

في جافا يوجد دالتين يمكنك استخدامهم للطباعة, ذكرناهم في الجدول التالي .

الدالة	تعريفها
System.out.print()	دالة تستخدم لعرض أي شيء نضعه في داخلها سواء نص, رقم أو قيمة متغير .
System.out.println()	نفس الدالة السابقة , الفرق بينها و بين الدالة السابقة أنها تعرض أي شيء نطبعه بعدها على سطر جديد .



- **مثلة شاملة حول دوال العرض :**

في المثال التالي قمنا بعرض ثلاث أشياء باستخدام الدالة **print()**.

المثال الأول

```
public class Main {  
  
    public static void main(String[] args) {  
  
        // هنا قمنا بعرض نص  
        System.out.print("Welcome to java world");  
  
        // هنا قمنا بعرض رقم  
        System.out.print(1000);  
  
        // هنا قمنا بتعريف متغير اسمه x بعدها قمنا بعرض قيمته  
        int x = 123;  
        System.out.print(x);  
  
    }  
  
}
```

سنحصل على النتيجة التالية عند التشغيل

Welcome to java world1000123

- إذا الدالة **print ()** تعرض أي شيء نضعه فيها .



❖ هنا قمنا بكتابة نفس البرنامج السابق , لكننا استخدمنا الدالة **println()** بدلاً من الدالة **print()**.

المثال الثاني

```
public class Main {  
  
    public static void main(String[] args) {  
  
        // هنا قمنا بعرض نص  
        System.out.println("Welcome to java world");  
  
        // هنا قمنا بعرض رقم  
        System.out.println(1000);  
  
        // هنا قمنا بتعريف متغير اسمه x بعدها قمنا بعرض قيمته  
        int x = 123;  
        System.out.println(x);  
  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

```
Welcome to java world  
1000  
123
```

- إذا الدالة **println()** تعرض أي شيء نضعه فيها, بالإضافة إلى ذلك تجعل أي شيء نعرضه بعدها ينزل على سطر جديد .

الوحدة الثالثة

الخوارزميات وخرائط التدفق

نظري

طريقة حل المشكلة Problem Solving ؟

١. فهم المشكلة :

حل المشكلة بعناية فائقة محاولا فهم كل جزئياتها وتحديد كل المتطلبات للحصول على الحل المقبول وفهم كل ما يؤدي للحصول على الحل المقبول للمشكلة , ملخص هذه القاعدة هو أن فهم المشكلة يمثل نصف الحل وكذلك الفهم الجيد والصحيح والكامل للمشكلة يعطي دائما نتائج واضحة وصحيح .

٢. تقسيم المشكلة :

حاول أن تقسم المشكلة إلى أجزاء بسيطة وغير معتمدة على بعضها البعض ثم ركز على كل جزء على حدة , والغرض من تقسيم المشكلة هو العمل مع جزء واحد فقط وعزل تأثير الأجزاء الأخرى حتى يسهل التعامل معها .

٣. حل المشكلة :

عند تقسيم المشكلة إلى أقسام صغيرة يجب أن يكون التقسيم على خطوات متعددة بحيث تستخدم القواعد العامة في المراحل الأولى ثم يتم الانتقال إلى المراحل الخاصة بعد ذلك , في كل مرحلة من المراحل يجب مراجعة الحل المقترح ليتم التأكد من أنه كامل وصحيح

الخوارزميات Algorithms ؟

هي مجموعة من الخطوات والتعليمات المرتبة , لتنفيذ عملية حسابية , أو منطقية , أو غيرها بشكل متتابع متسلسل ومنظم .

لقد استخدمت كلمة الخوارزمية في القرن الماضي وبشكل واسع في أوروبا وأمريكا , وكانت تعني الوصف الدقيق لتنفيذ مهمة من المهمات أو حل مسألة من المسائل , وقد اشتق الغربيون هذه الكلمة من اسم عالم الرياضيات المسلم المعروف ,

محمد بن موسى الخوارزمي .



صيغة الخوارزميات !

- ❖ انا جوعان , تعتبر هذه مشكلة .
- ❖ ما ذا افعل لحل هذه المشكلة ؟
- ✓ الخطوة الأولى : الفلوس .
- ✓ الخطوة الثانية : الذهاب الى المطعم .
- ✓ الخطوة الثالثة : اطلب .
- ✓ الخطوة الرابعة : تناول الوجبة .
- ❖ هذه الاربعة الخطوات حلت هذه المشكلة .


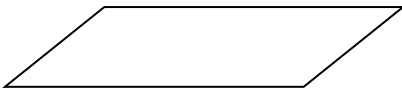
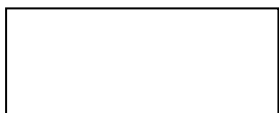
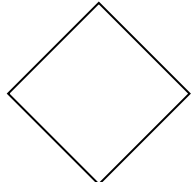
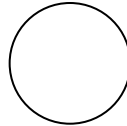
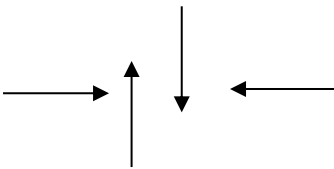
خرائط التدفق Flow Charts ؟

وهي وسيلة بصرية مفيدة للتعبير عن الخطوات المنطقية اللازمة لحل مسألة ما ,أو يمكن القول بأنها تمثيل بياني تخطيطي لخطوات حل مسألة معينة .

❖ أهمية استخدام خرائط سير العمليات :

- i. تعطي صورة كاملة لخطوات حل المسألة .
- ii. تساعد المبرمج على تشخيص الأخطاء التي تقع عادة في البرنامج .
- iii. تيسر للمبرمج عملية متابعة المسائل و الاحتمالات والتفرعات .
- iv. تيسر للمبرمج إدخال التعديلات على البرنامج .
- v. تعتبر رسوم خرائط سير العمليات مرجعاً لحل المسائل المتشابهة .

أشكال خرائط التدفق Flow Charts ؟

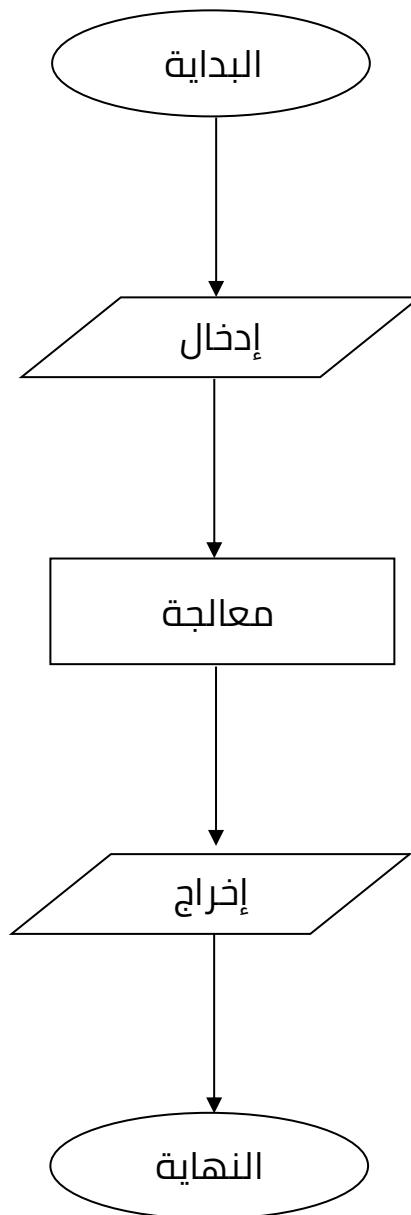
الشكل	الإستخدام
	البداية أو النهاية (Terminal)
	الإدخال أو الإخراج (Input / Output)
	معالجة أو عملية (Process)
	اتخاذ قرار (Decision)
	نقطة توصيل وربط (Connector)
	خطوط إتجاه (Flow Lines)

أنواع خرائط سير البرامج ؟

١. خرائط التتابع البسيط Simple Sequential Flowcharts :

ويتم ترتيب خطوات الحل لهذا النوع من الخرائط، بشكل سلسلة مستقيمة، من بداية البرنامج حتى نهايته، بحيث تنعدم فيها أية تفرعات على الطريق، كما تخلو من أي دورانات مما هو موجود في الأنواع الأخرى من الخرائط ،

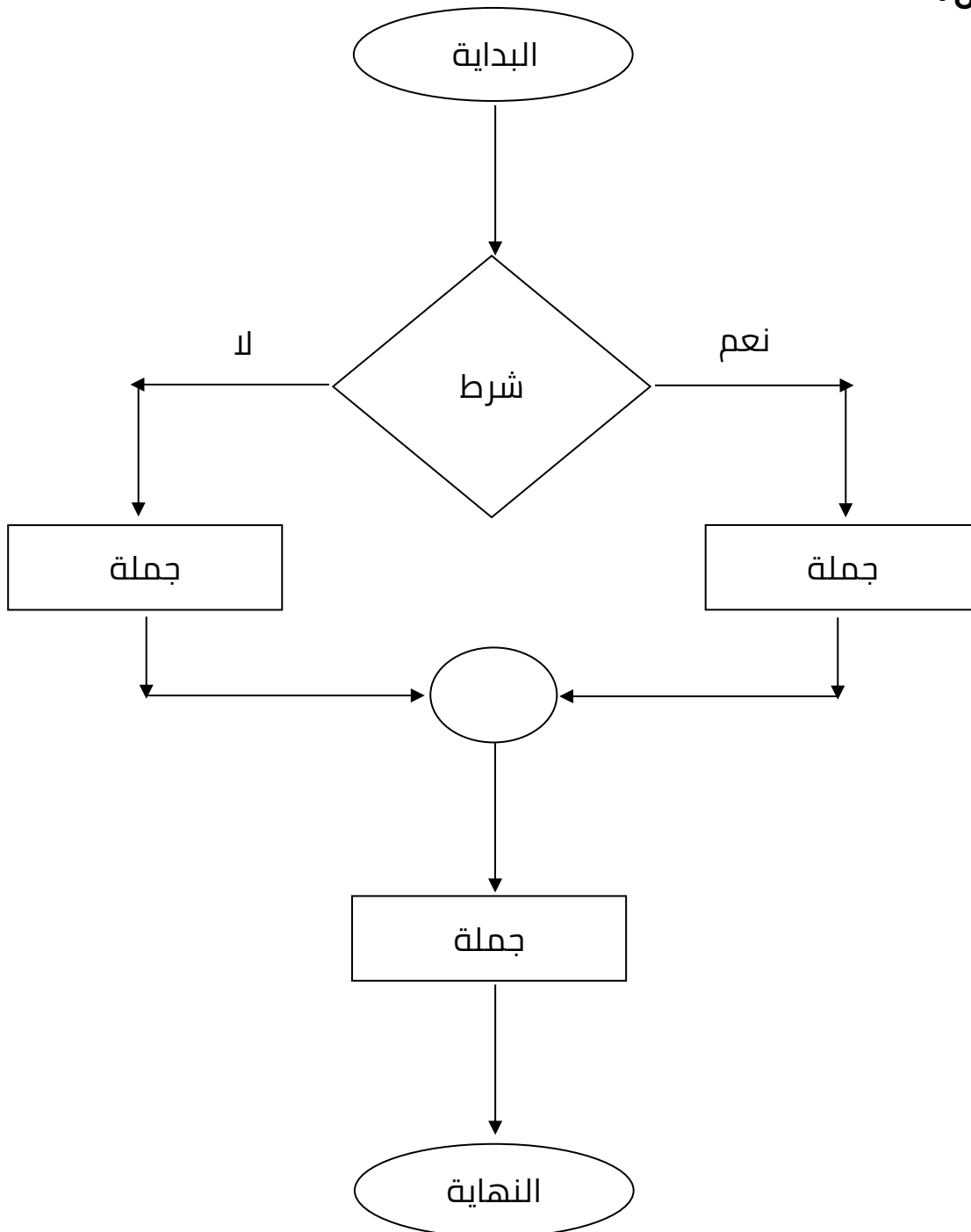
مثال :



٢. الخرائط ذات الفروع Branched Flowcharts :

إن أي تفرع يحدث في البرنامج، إنما يكون بسبب الحاجة لاتخاذ قرار، أو مفاضلة بين اختيارين أو أكثر، فيسير كل اختيار في طريق مستقل (تفرع) عن الآخر.

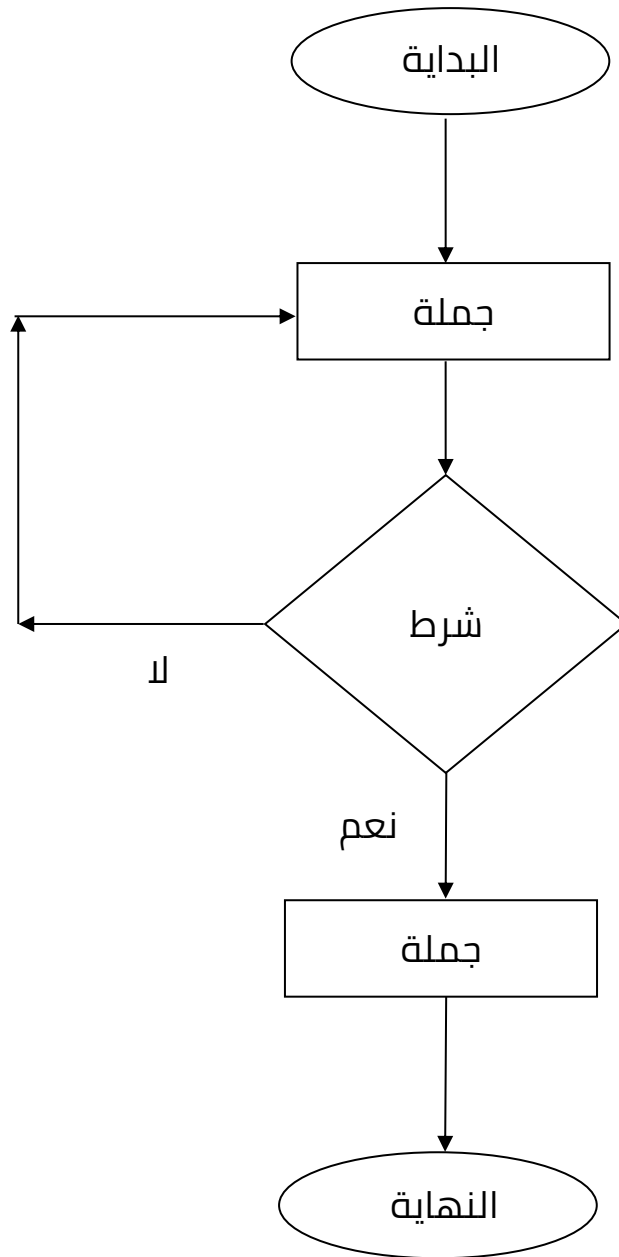
مثال :



٣. خرائط الدوران الواحد Simple – Loop Flowcharts :

وهذه الخرائط نحتاج اليها لإعادة عملية أو مجموعة من العمليات في البرنامج عددا محدودا أو غير محدود من المرات , وقد سميت هذه الخرائط بخرائط الدوران الواحد لأنها تستعمل حلقة واحدة, وتسمى أحيانا خرائط الدوران البسيط .

مثال :



الوحدة الثالثة

الخوارزميات وخرائط التدفق

عملي

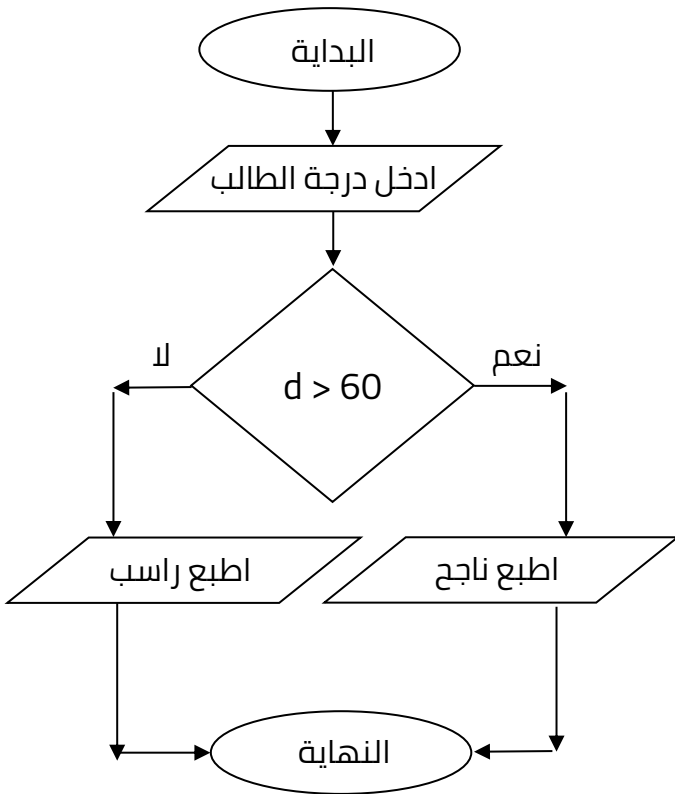
أولاً : خرائط التتابع البسيط

مثال / قم بكتابة الخوارزميات وخرائط التدفق لبرنامج يقوم بجمع عددين ومن ثم طباعة الناتج النهائي ؟

طريقة خرائط التدفق	طريقة الخوارزميات
<pre> graph TD Start([البداية]) --> Input1[/ادخل الرقم الأول/] Input1 --> Input2[/ادخل الرقم الثاني/] Input2 --> Process[المجموع = الرقم ١ + الرقم ٢] Process --> Output[/اطبع الناتج/] Output --> End([النهاية]) </pre>	<ul style="list-style-type: none"> • البداية . • ادخل الرقم الأول Num1 • ادخل الرقم الثاني Num2 • المجموع = الرقم الأول + الرقم الثاني • $Sum = Num1 + Num2$ • اطبع الناتج Print Sum • النهاية

ثانياً : الخرائط ذات الفروع

مثال / قم بكتابة الخوارزميات وخرائط التدفق لبرنامج يقوم بقراءة درجة الطالب في الاختبار فإذا كانت درجته أكبر من أو يساوي ٦٠ يخرج كلمة (ناجح) و إذا كانت أقل من ذلك يخرج البرنامج كلمة (راسب) ؟

طريقة خرائط التدفق	طريقة الخوارزميات
 <pre> graph TD Start([البداية]) --> Input[/ادخل درجة الطالب/] Input --> Decision{d > 60} Decision -- نعم --> PrintPass[/اطبع كلمة ناجح/] Decision -- لا --> PrintFail[/اطبع راسب/] PrintPass --> End([النهاية]) PrintFail --> End </pre>	<ul style="list-style-type: none"> • البداية • ادخل درجة الطالب input d • قم بمقارنة الدرجة مع 60 • إذا كانت الدرجة أكبر من أو يساوي 60 اطبع كلمة ناجح • إذا كانت الدرجة أصغر من 60 اطبع كلمة راسب • النهاية



ثالثاً : خرائط الدوران الواحد

مثال / قم بكتابة الخوارزميات وخرائط التدفق لشخص يريد الجلوس على كرسي يبعد عنه عدة أمتار ؟

طريقة خرائط التدفق	طريقة الخوارزميات
<pre>graph TD Start([البداية]) --> Step1[تقدم خطوة] Step1 --> Decision{وصلت للكرسي؟} Decision -- لا --> Step1 Decision -- نعم --> Step2[اجلس على الكرسي] Step2 --> End([النهاية])</pre>	<ul style="list-style-type: none">• البداية• تقدم خطوة• اذا لم تصل الى الكرسي انتقل الى الخطوة رقم 2• اذا وصلت الى الكرسي انتقل الى الخطوة رقم 4• اجلس على الكرسي• النهاية

الوحدة الرابعة

المتغيرات وإسناد القيم

نظري



مفهوم المتغيرات variables ؟

هو مكان يتم حجزه في الذاكرة لتخزين بيانات أثناء تشغيل البرنامج , النوع الذي نعطيه للمتغير يجعل نظام التشغيل يحدد نوع البيانات الذي يمكن تخزينه في المساحة المحجوزة لهذا المتغير في الذاكرة .

أنواع المتغيرات ؟

١. Integer متغير رقمي :

- هذا النوع يمثل عدد صحيح يتألف من 32 bit .
- أقل قيمة يمكن تخزينها فيه -٢,١٤٧,٤٨٣,٦٤٧ .
- أكثر قيمة يمكن تخزينها فيه +٢,١٤٧,٤٨٣,٦٤٦ .
- إذا لم نضع أي قيمة , توضع القيمة 0 كقيمة افتراضية .
- اختصاره int ويستخدم لتخزين عدد صحيح .

مثال

```
int a ;  
a = 5 ;
```

٢. String متغير نصي :

- يستخدم لتعريف النصوص .
- ويتم وضع القيمة داخل علامتي تنصيص " "

مثال

```
String a ;  
a = "Ahmed" ;
```

٣. Long

- هذا النوع يمثل عدد صحيح يتألف من 64 bit.
- أقل قيمة يمكن تخزينها فيه هي -٨٠٨,٧٧٥,٨٥٤,٣٦,٣٧٢,٩٢٣.
- أكثر قيمة يمكن تخزينها فيه هي +٨٠٧,٧٧٥,٨٥٤,٣٦,٣٧٢,٩٢٣.
- إذا لم نضع أي قيمة , توضع القيمة ٠ كقيمة افتراضية .
- النوع long يستخدم لتخزين عدد كبير جداً لا يحتوي على فاصلة عشرية, أي لتخزين عدد صحيح حجمه كبير جداً .

مثال

```
long a = 1234567L ;
long b = -700000L ;
```

٤. Char

- هذا النوع يمثل معلومة تتألف من ١٦ bit.
- يخزن حرف او رمز واحد فقط .
- يتم وضع القيمة داخل علامتي تنصيص مفردة ' ' .
- أقل قيمة يمكن تخزينها فيه هي ٠.
- أكثر قيمة يمكن تخزينها فيه هي ١٠,٠٣٥.

مثال

```
Char a ;
a = ' B ' ;
```

0. Boolean

- هذا النوع يمثل معلومة تتألف من 1 bit.
- يستطيع أن يحتوي إما على القيمة true أو على القيمة false.
- إذا لم نضع أي قيمة، توضع القيمة false كقيمة افتراضية.
- النوع boolean يستخدم في الشروط.

مثال

```
boolean check = true ;  
boolean found = false ;
```

1. Float

- هذا النوع يمثل عدد بفاصلة عشرية يتألف من 32 bit.
- إذا لم نضع أي قيمة، توضع القيمة f٠,٠ كقيمة افتراضية.
- النوع float يستخدم لتخزين عدد كبير بفاصلة عشرية.

مثال

```
float a = 12.05f ;  
float b = - 8.123f ;
```

Double .V

- هذا النوع يمثل عدد بفاصلة عشرية يتألف من 64 bit .
- إذا لم نضع أي قيمة , توضع القيمة 0.0 كقيمة افتراضية .
- النوع double يستخدم لتخزين عدد كبير جداً بفاصلة عشرية .

مثال

```
double a = 50.98794d ;  
double b = -100.1d ;
```

طريقة إنشاء المتغيرات ؟

```
int a ;
```

عليك أولاً التصريح عن متغير ثم تخزين القيم فيه

```
a = 100 ;
```

إعطاء المتغير قيمة

```
System.out.println (a)
```

طباعة المتغيرات

الوحدة الرابعة

المتغيرات وإسناد القيم

عملي

١. Integer متغير رقمي :

مثال / كتابة برنامج يقوم بجمع عددين , العدد الأول = ٣٠ والعدد الثاني = ٢٠
ثم يقوم بطباعة مجموعهما :

المثال

```
public class Main {

    public static void main(String[] args) {

        int Num1 ;
        int Num2 ;
        int sum ;

        Num1 = 30 ;
        Num2 = 20 ;

        sum = Num1 + Num2 ;

        System.out.println( "sum = " +sum ) ;

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

Sum = 50

- أولاً تم تعريف المتغير من نوع integer لأنه رقمي .
- ثانياً تم إسناد قيم لهذه المتغيرات .
- ثالثاً تم إجراء عملية حسابية بسيطة بإستخدام دالة Sum
- رابعاً تم طباعة الناتج وهو حاصل جمع العددين

٢. String متغير نصي :

مثال / كتابة برنامج يكون الاسم الأول = "Mohammed" والاسم الثاني = "Fahad" ويقوم بطباعة كل اسم في سطر جديد :

المثال

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String Name1 ;  
        String Name2 ;  
  
        Num1 = "Mohammed" ;  
        Num2 = "Fahad" ;  
  
        System.out.println(Num1) ;  
        System.out.println(Num2) ;  
  
    }  
  
}
```

سنحصل على النتيجة التالية عند التشغيل

```
Mohammed  
Fahad
```

- أولاً تم تعريف المتغير من نوع String لأنه نصي .
- ثانياً تم إسناد قيم لهذه المتغيرات .
- ثالثاً تم طباعة كل اسم في سطر جديد .

الوحدة الخامسة

الطباعة والادخال من خلال JOptionPane

نظري



الكلاس JOptionPane

يستخدم لإظهار نافذة خيارات (Option Pane) أمام المستخدم لعدة أسباب :

١. لتحذيره أو إعلامه بشيء ما .
٢. لسؤاله إذا كان موافقاً على أمر ما أم لا .
٣. ليطالب منه بإدخال قيمة .

❖ الكلاس JOptionPane يوفر دوال كثيرة يمكن إستخدامها للحصول على :

Message Dialog ✓

Confirm Dialog ✓

Input Dialog ✓

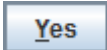
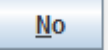
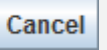
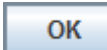
المصطلح	معناه
Message Dialog	عبارة عن نافذة منبثقة تستخدم لإعلام المستخدم بشيء ما . نحصل على Message Dialog بواسطة الدالة (<code>showMessageDialog()</code>)
Confirm Dialog	عبارة عن نافذة منبثقة تستخدم لسؤال المستخدم إذا كان موافقاً على شيء ما أم لا . نحصل على Confirm Dialog بواسطة الدالة (<code>showConfirmDialog()</code>)
Input Dialog	عبارة عن نافذة منبثقة تستخدم لجعل المستخدم يقوم بإدخال أو اختيار قيمة ما . نحصل على Input Dialog بواسطة الدالة (<code>showInputDialog()</code>)

ثوابت الكلاس JOptionPane

الجدول يحتوي على ثوابت الكلاس JOptionPane المخصصة لتحديد أيقونة الـ Option Pane

الثابت	استخدامه	الأيقونة
ERROR_MESSAGE	يظهر أيقونة تلفت نظر المستخدم إلى وجود خطأ ما	
INFORMATION_MESSAGE	يظهر أيقونة تلفت نظر المستخدم لقراءة معلومة ما	
WARNING_MESSAGE	يظهر أيقونة تلفت نظر المستخدم لوجود تحذير ما	
QUESTION_MESSAGE	يظهر أيقونة تلفت نظر المستخدم لسؤاله عن شيئاً ما	
PLAIN_MESSAGE	يستخدم لعدم إظهار أي أيقونة	

الجدول التالي يحتوي على ثوابت الكلاس JOptionPane المخصصة لتحديد الأزرار التي ستوجد في الـ Option Pane و تحديداً في الـ Confirm Dialog

الثابت	إستخدامه	الأزرار
YES_NO_OPTION	يستخدم لإظهار الأزرار Yes و No في الـ Option Pane	 
YES_NO_CANCEL_OPTION	يستخدم لإظهار الأزرار Yes و No و Cancel في الـ Option Pane	  
OK_CANCEL_OPTION	يستخدم لإظهار الأزرار OK و Cancel في الـ Option Pane	 

إخراج البيانات

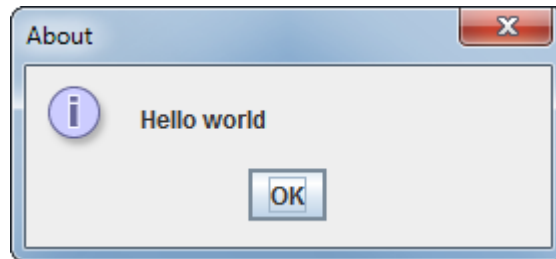
❖ لغة الجافا تحتوي على أكثر من طريقة لإخراج أو (طباعة) البيانات :

١- باستخدام **system.out.println** (تستخدم للتطبيقات غير الرسومية) .

مثال

```
System.out.println ("Hello World") ;
```

٢- باستخدام **JOptionPane** (يتم استخدامها للتطبيقات الرسومية) .



طريقة إدخال البيانات باستخدام JOptionPane :

أولاً : يجب تحميل الكائن أو الكلاس المسمى JOptionPane من الحزمة المسماة javax.swing ويتم ذلك عن طريق الكود التالي :

مثال

```
import javax.swing.JOptionPane ;
```



ثانيا : يجب تعريف المتغيرات التى سيتم تخزين ما يقوم المستخدم بإدخاله عن طريق لوحة المفاتيح ولابد أن تكون من النوع النصي String وذلك لان كل بيانات يتم إدخالها من المستخدم بواسطة هذا الأمر يعتبر String

مثال

```
String name ;
```

ثالثا : نقوم بإظهار المربع الحوار لادخال البيانات من المستخدم ويتم تخزينها فى المتغير String

مثال

```
name=JOptionPane.showInputDialog("من فضلك أدخل أسمك");
```

رابعا : إنهاء البرنامج وذلك بإستخدام الامر

مثال

```
System.exit( 0 ) ;
```

✓ و كقاعدة عامة يجب أستخدام هذا الامر لانهاء التطبيق فى التطبيقات الرسومية وذلك لان التطبيق فى حالة أستخدام هذا الامر سيكون نشط فى ذاكرة الجهاز .

- الكود العام لادخال البيانات باستخدام الكلاس JOptionPane سيكون كالتالي :

مثال

```
package m;
import javax.swing.JOptionPane ;
Public class m {
    Public static void main (String[] args) {
        String name ;
        name = JOptionPane.showInputDialog; ("من فضلك أدخل اسمك")
        System.exit( 0 ) ;
    }
}
```

انتهينا من كيفية ادخال البيانات النصية , لكن ماذا لو كنا نريد من المستخدم إدخال بيانات رقمية و ليست نصية و نريد استخدامها كأرقام في عمليات حسابية ؟

فقد أخبرنا سابقا بأن كل شيء يتم أخذه من المستخدم بواسطة هذا الأمر (Input Dialog) يعتبر String بمعنى أن البرنامج سيعتبرها نصوص, ولذلك لن يتمكن من التعامل معها في المعادلات الرياضية والحسابية حتى لو أدخل المستخدم فيها أرقام .

بمعنى لو أدخل المستخدم رقم سيتم تخزينه في البرنامج بين علامتي التنصيص بهذا الشكل: "٩٩"

أن قمنا بعمليات حسابية لها مباشرة , سيحدث خطأ في البرنامج , لأنه لن يستطيع التعامل مع ال String في العمليات الحسابية.

لذلك توجد أوامر لتحويل المدخلات من بيانات نصية إلى أنواع البيانات الرقمية التي نريدها.



❖ تحويل القيم النصية إلى أرقام :

مثال

```
(متغير رقمي) Integer.parseInt = متغير نصي
```

طريقة إخراج البيانات باستخدام JOptionPane :

طريقة إخراج البيانات باستخدام الكلاس المسمى **JOptionPane** كنا قد تعودنا على استخدامها في الدروس السابقة وقلنا أننا إذا أردنا أن نخرج بيانات للمستخدم على هيئة نافذة رسالة يكون عن طريق جملة الاخراج التالي :

مثال

```
JOptionPane.showMessageDialog(null, "");
```

وكانت هذه ابسط رسالة يمكن اظهارها للمستخدم و تحتوي بيانات , لكن يمكننا اظهارها بصورة أكثر تنسيق باستخدام جملة الاخراج التالية :

مثال

```
showMessageDialog(null, message, title, messageType);
```

❖ وجملة الاخراج السابقة فكما هو واضح تتطلب أربع مدخلات :

المدخل الاول : الكلمة المحبوزة **null** وفائدتها وضع صندوق الحوار في وسط الشاشة .



المدخل الثاني : **message** نص الرسالة الذي سوف يظهر للمستخدم ويجب وضع علامتي اقتباس على النص المراد إخراجها للمستخدم .

المدخل الثالث : **title** عنوان الرسالة الذي سوف يظهر في سطر العنوان للرسالة .

المدخل الرابع : **messageType** وهو نوع الرسالة حيث يوجد مجموعة من الرموز التي يمكن اظهارها في صندوق الحوار التي تساعد المستخدم في معرفة نوع صندوق الحوار و الرسالة التي تظهر فيه .

أنواع العمليات (Operations Types) :

أولاً : العمليات الإسنادية (Assignment Operations)

✓ تستخدم هذه العمليات لإسناد قيمة معينة لمتغير ما بعد تعريف هذا المتغير .

مثال

```
X = 1  
radius = 1.0  
ch = 'A'
```

○ ملاحظات مهمة :

✓ لا تسند قيمة من نوع يختلف عن نوع المتغير كأن تسند قيمة صحيحة لمتغير من نوع char مثلاً .

✓ اسم المتغير دائماً يكون على يسار الإشارة وقيمة المتغير تكون على يمين الإشارة، أي أن الصيغة التالية تعتبر خطأ (X = 1) .

❖ يمكن كتابة عمليات الإسناد بصورة مختصرة كما هو موضح بالجدول التالي :

العامل	مثال	ما يقابله دون اختصار
$+=$	$c += 7$	$c = c + 7$
$-=$	$d -= 4$	$d = d - 4$
$*=$	$e *= 5$	$e = e * 5$
$/=$	$f /= 3$	$f = f / 3$
$\%=$	$g \% = 9$	$g = g \% 9$

ثانيا : **العمليات الحسابية** (Arithmetic Operations) :

✓ معظم برامج الحاسوب تقوم بعمليات حسابية , والشكل التالي يوضح هذه العمليات :

العملية	العامل	التعبير الجبري	التعبير بالجافا
جمع	$+$	$f + 7$	$f + 7$
طرح	$-$	$f - 7$	$f - 7$
ضرب	$*$	bm	$b * m$
قسمة	$/$	X	x / y
موديلاس	$\%$	$r \bmod s$	$r \% s$

❖ **عامل الزيادة والنقصان** (Increment & Decrement) :

تمدنا لغة جافا بعامل الزيادة ++ وعامل النقصان -- وذلك من أجل زيادة قيمة متغير أو إنقاص قيمته بمقدار واحد .

✓ $++a$ بدلا من $a = a + 1$ أو بدلا من $a += 1$

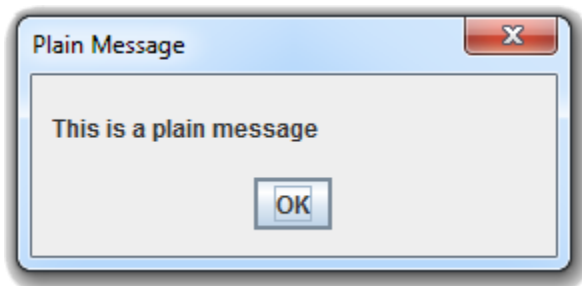
✓ $--a$ بدلا من $a = a - 1$ أو بدلا من $a -= 1$

الوحدة الخامسة

الطباعة والادخال من خلال JOptionPane

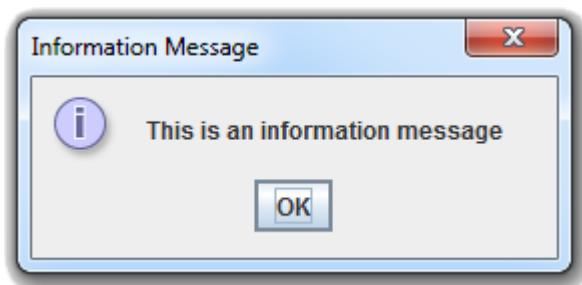
عملي

أشكال ال JOptionPane :



السطر البرمجي

```
JOptionPane.showMessageDialog(null, "This is a plain  
message", "Plain Message", JOptionPane.PLAIN_MESSAGE);
```



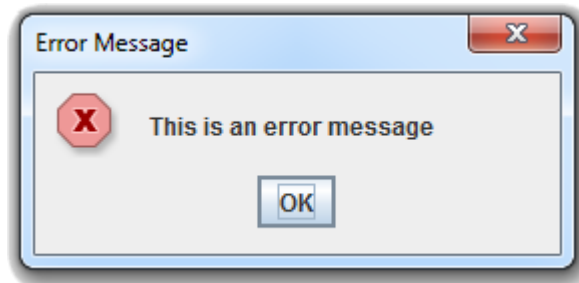
السطر البرمجي

```
JOptionPane.showMessageDialog(null, "This is an information  
message", "Information Message",  
JOptionPane.INFORMATION_MESSAGE);
```



السطر البرمجي

```
JOptionPane.showMessageDialog(null, "This is a warning  
message", "Warning Message", JOptionPane.WARNING_MESSAGE);
```



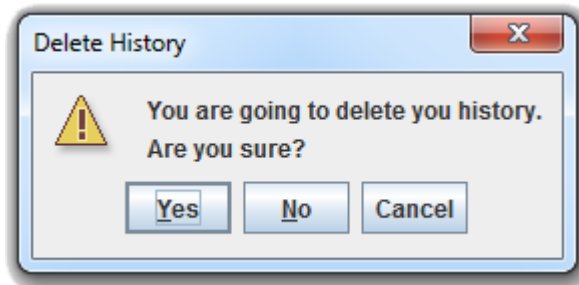
السطر البرمجي

```
JOptionPane.showMessageDialog(null, "This is an error  
message", "Error Message", JOptionPane.ERROR_MESSAGE);
```



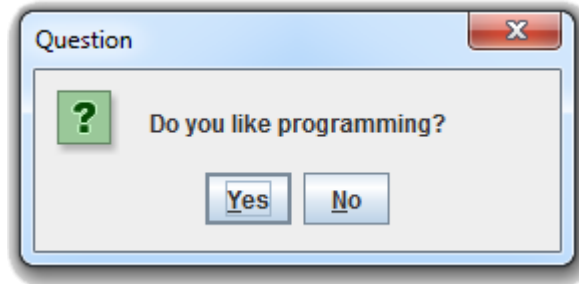
السطر البرمجي

```
JOptionPane.showMessageDialog(null, "This is a question  
message", "Quesiton Message", JOptionPane.QUESTION_MESSAGE);
```



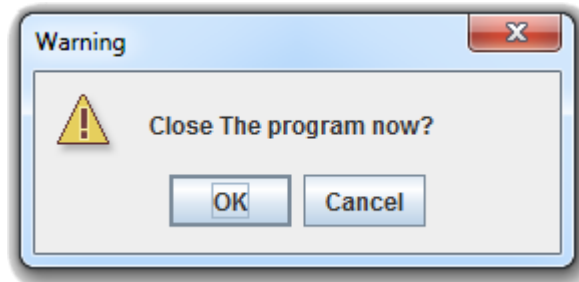
السطر البرمجي

```
JOptionPane.showConfirmDialog(null,  
"You are going to delete you history.\nAre you sure?","Delete History",  
JOptionPane.YES_NO_CANCEL_OPTION,  
JOptionPane.WARNING_MESSAGE);
```



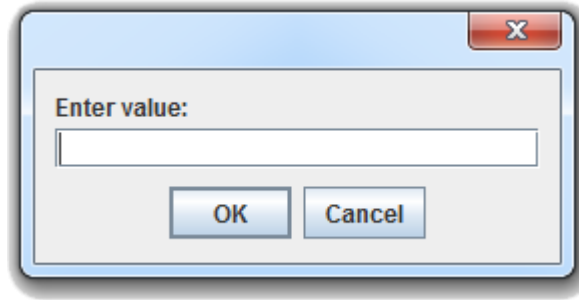
السطر البرمجي

```
JOptionPane.showConfirmDialog(null,"Do you like programming?","Quesiton",
JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
```



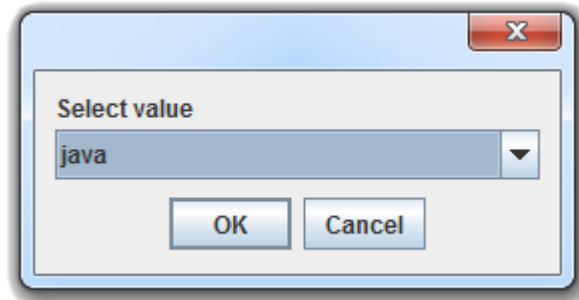
السطر البرمجي

```
JOptionPane.showConfirmDialog(null,"Close The program now?","Warning",
JOptionPane.OK_CANCEL_OPTION,
JOptionPane.WARNING_MESSAGE);
```



السطر البرمجي

```
JOptionPane.showInputDialog(null, "Enter value:", "",
JOptionPane.PLAIN_MESSAGE);
```



السطر البرمجي

```
Object[] values = {"java", "css", "c++", "android"};

JOptionPane.showInputDialog(null, "Select value", "", JOptionPane.PLAIN_MESSAGE,
null, values, values[0]);
```

الوحدة السادسة

التعبيرات المنطقية وكتابة جملة IF , else

نظري



مفهوم الشرط في الجافا :

تعني conditions في اللغة الإنجليزية , و نستخدم الشروط لتحديد طريقة عمل البرنامج .
كما أنه يمكنك وضع العدد الذي تريده من الشروط في البرنامج الواحد , تستطيع وضع الشروط بداخل بعضها البعض أيضاً .

❖ أنواع جمل الشرط :

١. جمل الشرطية		
جملة if	جملة else	
جملة else if	جملة switch	
٢. جمل الدوران		
جملة while	جملة do-while	جملة for

❖ جمل الشرط بشكل عام :

مثال

```

if ( condition )
{
    // إذا كان الشرط صحيحاً نفذ هذا الكود
}

else if ( condition )
{
    // إذا كان الشرط صحيحاً نفذ هذا الكود
}

else
{
    // نفذ هذا الكود في حال لم يتم التعرف على الكود في أي شرط
}

```


العوامل التي تستخدم في وضع الشروط المنطقية :

AND - OR				مثال	رمزه	اسم العامل
A	B	A B	A && B	(a && b)	&&	AND
True	True	True	True	(a b)		OR
True	False	True	False			
False	True	True	False			
False	False	False	False	!a	!	NOT

العوامل التي تستخدم في المقارنات :

شرح الكود	مثال	رمزه	اسم العامل
هل قيمة a تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع true	(a == b)	==	يساوي
هل قيمة a لا تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع true	(a != b)	!=	لا يساوي
هل قيمة a أكبر من قيمة b ؟ إذا كان الجواب نعم فإنها ترجع true	(a > b)	>	أكبر من
هل قيمة a أصغر من قيمة b ؟ إذا كان الجواب نعم فإنها ترجع true	(a < b)	<	أصغر من
هل قيمة a أكبر أو تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع true	(a >= b)	>=	أكبر من أو يساوي
هل قيمة a أصغر أو تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع true	(a <= b)	<=	أصغر من أو يساوي



١. العامل && (AND Operator) :

- العامل && يستخدم لتنفيذ كود معين إذا تحقق الشرط الأول و الشرط الثاني .
- إذا كانت نتيجة الشرط الأول تساوي true و نتيجة الشرط الثاني تساوي true سينفذ الكود .
 - إذا لم تكن نتيجة كلا الشرطين تساوي true لن ينفذ الكود .

مثال

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 20;  
  
        // الشرط التالي يعني أنه إذا كانت قيمة المتغير a تساوي ١٠، و قيمة المتغير b تساوي ٢٠ سيتم تنفيذ أمر الطباعة  
        if( a == 10 && b == 20 )  
        {  
            System.out.println("The first and the second conditions return true");  
        }  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

The first and the second conditions return true

- نلاحظ أنه نفذ أمر الطباعة لأن جواب الشرطين الموضوعين في الجملة if هو true .



٢. العامل || (OR Operator) :

العامل || يستخدم لتنفيذ كود معين إذا تحقق على الأقل واحد من الشروط الموضوعة .
إذاً هنا يكفي أن يرجع أحد الشرطين القيمة true حتى يتم تنفيذ الأوامر الموضوعة .

مثال

```
public class Main {  
  
    public static void main(String[] args) {  
  
        int a = 10;  
        int b = 20;  
  
        // الشرط التالي يعني أنه إذا كانت قيمة المتغير a تساوي ١٠، أو قيمة المتغير b تساوي ٥٠ سيتم تنفيذ أمر الطباعة //  
  
        if( a == 10 || b == 50 )  
        {  
  
            System.out.println("One of the conditions return true");  
  
        }  
  
    }  
  
}
```

سنحصل على النتيجة التالية عند التشغيل

One of the conditions return true

- نفذ أمر الطباعة لأن جواب الشرط الأول الموضوع في الجملة if هو true .

٣. العامل ! (NOT Operator) :

العامل ! يستخدم لتنفيذ كود معين إذا لم يتحقق أي شرط تم وضعه .

إذا رجع الشرط أو جميع الشروط الموضوعية القيمة false سيتم تنفيذ الأوامر الموضوعية .

مثال

```
public class Main {  
    public static void main(String[] args) {  
        int a = 10;  
        // الشرط التالي يعني أنه إذا كانت قيمة a لا تساوي ٢٠ سيتم تنفيذ أمر الطباعة  
        If ( ! ( a == 20 ) )  
        {  
            System.out.println("The condition return true");  
        }  
    }  
}
```

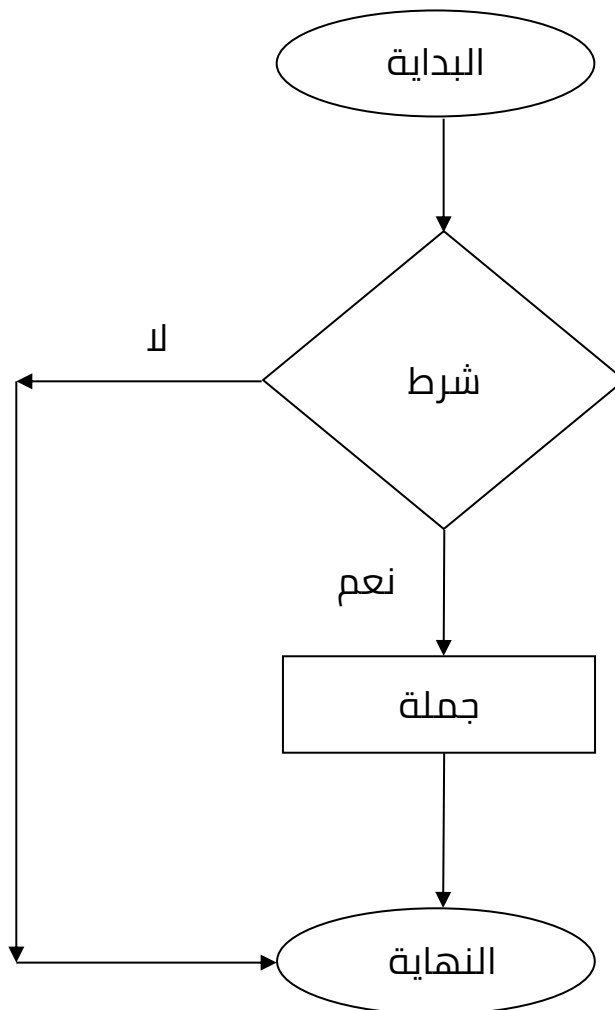
سنحصل على النتيجة التالية عند التشغيل

The condition return true

- نفذ أمر الطباعة لأن جواب الشرط هو false .

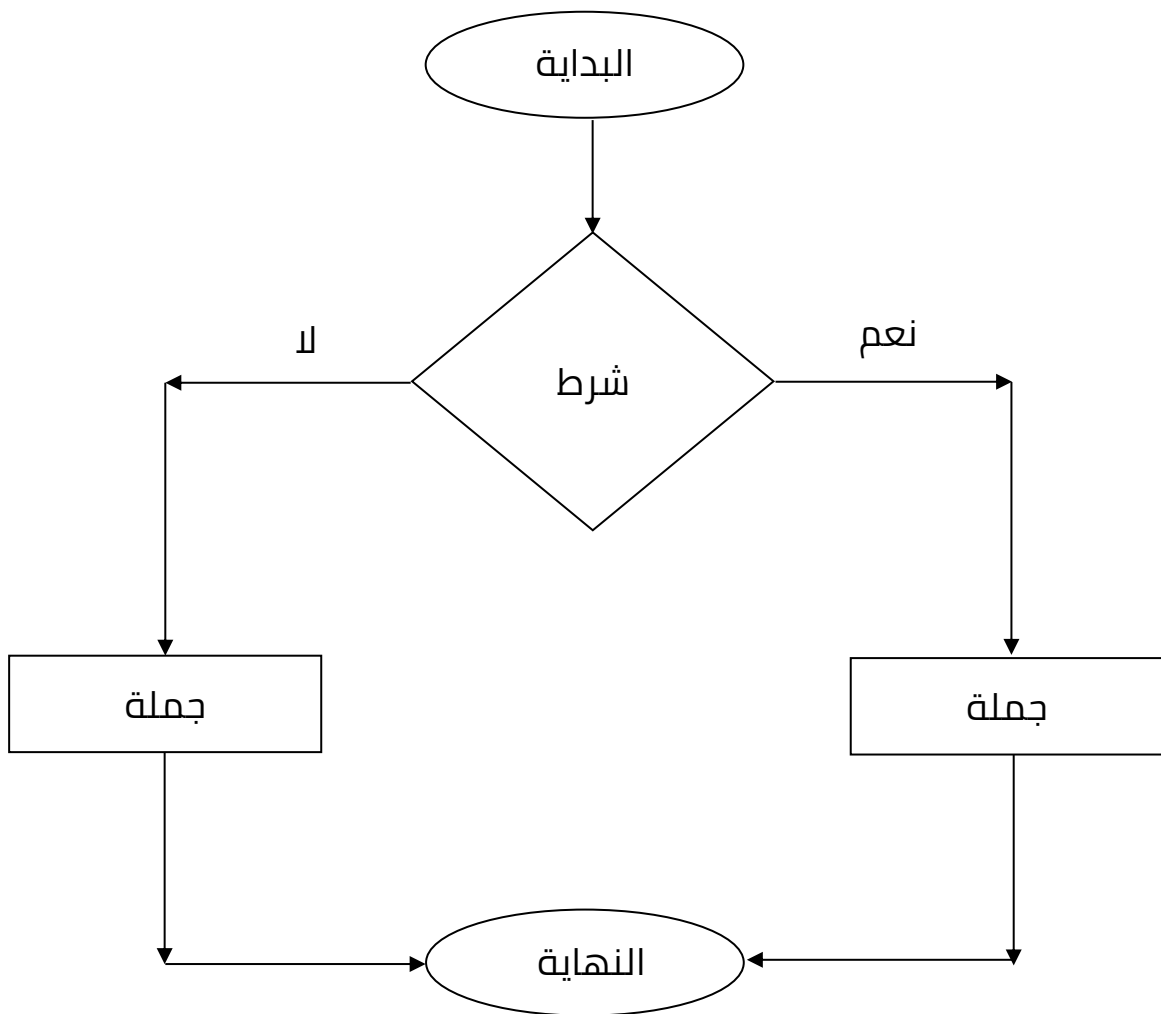
**أولاً : جملة if**

- أبسط الجمل الشرطية فهي تحتوي على شرط , عند تحققه ينفذ أمر معين .
- تستخدم اذا كان هناك شرط واحد فقط .
- if في اللغة العربية تعني " إذا " .
- الجملة الشرطية IF هي أحد أهم الأدوات في البرمجة بشكل عام , و عن طريقها نستطيع تحديد ما اذا كانت حالة معينة صحيحة True أو خاطئة False .

❖ طريقة عمل جملة if :

ثانياً : جملة else

- تستخدم اذا كان هناك شرطين .
- عند عدم تحقيق الشرط ينفذ أمر معين .
- else في اللغة العربية تعني " غير ذلك " .



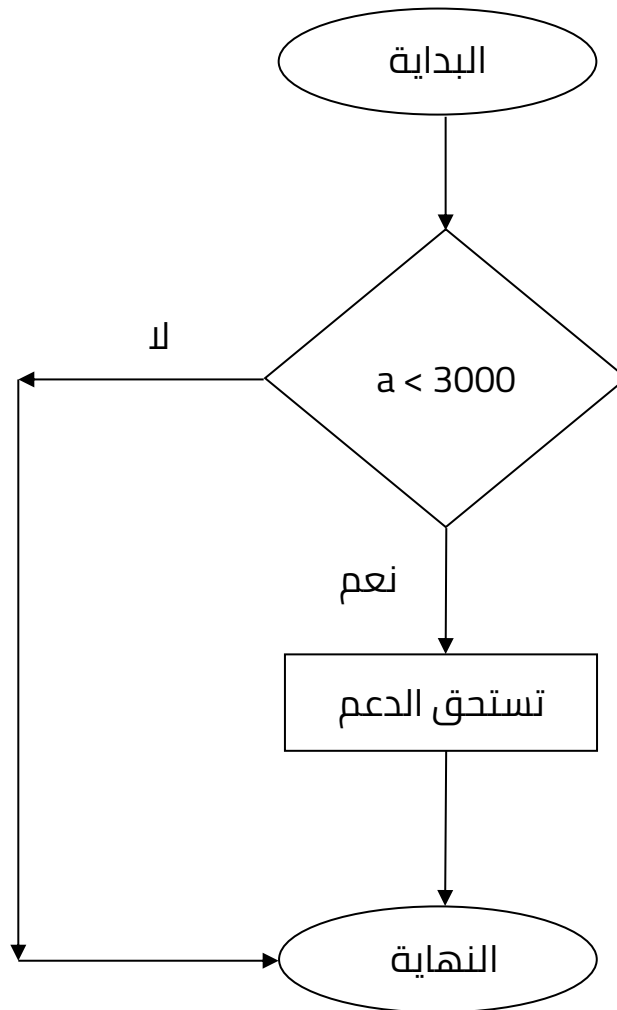
الوحدة السادسة

التعبيرات المنطقية و كتابة جملة IF

عملي

❖ جملة if :

قم بكتابة برنامج يقوم المستخدم بإدخال الراتب اذا كان الراتب أقل من ٣٠٠٠ يقوم بطباعة العبارة التالية : «تستحق الدعم»



- نلاحظ في جملة ال if اذا كان الراتب أقل من ٣٠٠٠ يتم طباعة جملة (تستحق الدعم)

وإذا كان الراتب أعلى من ٣٠٠٠ سيتم إنهاء البرنامج دون طباعة أي عبارة .

❖ خطوات الحل :

✓ جلب المكتبة لعرض رسالة الحوار

السطر البرمجي

```
import javax.swing.JOptionPane ;
```

✓ تعريف المتغيرات النصية والرقمية

السطر البرمجي

```
String a ;  
Int b ;
```

✓ إعطاء المتغيرات النصية كود الادخال

السطر البرمجي

```
a = JOptionPane.showInputDialog(null, "ادخل الراتب");
```

✓ عملية تحويل المتغيرات النصية إلى رقمية

السطر البرمجي

```
b=Integer.parseInt(a) ;
```

✓ استخدام دالة if

السطر البرمجي

```
if (b <= 3000)
```



السطر البرمجي

```
JOptionPane.showMessageDialog(null, "تستحق الدعم");
```

الكود العام

```
package Test;
import javax.swing.JOptionPane ;
public class Test {

    public static void main(String[] args) {

        String a ;
        a = JOptionPane.showInputDialog(null, "ادخل الراتب");

        int b ;
        b=Integer.parseInt(a) ;

        if (b <= 3000)

        {
            JOptionPane.showMessageDialog(null, "تستحق الدعم");
        }

    }

}
```

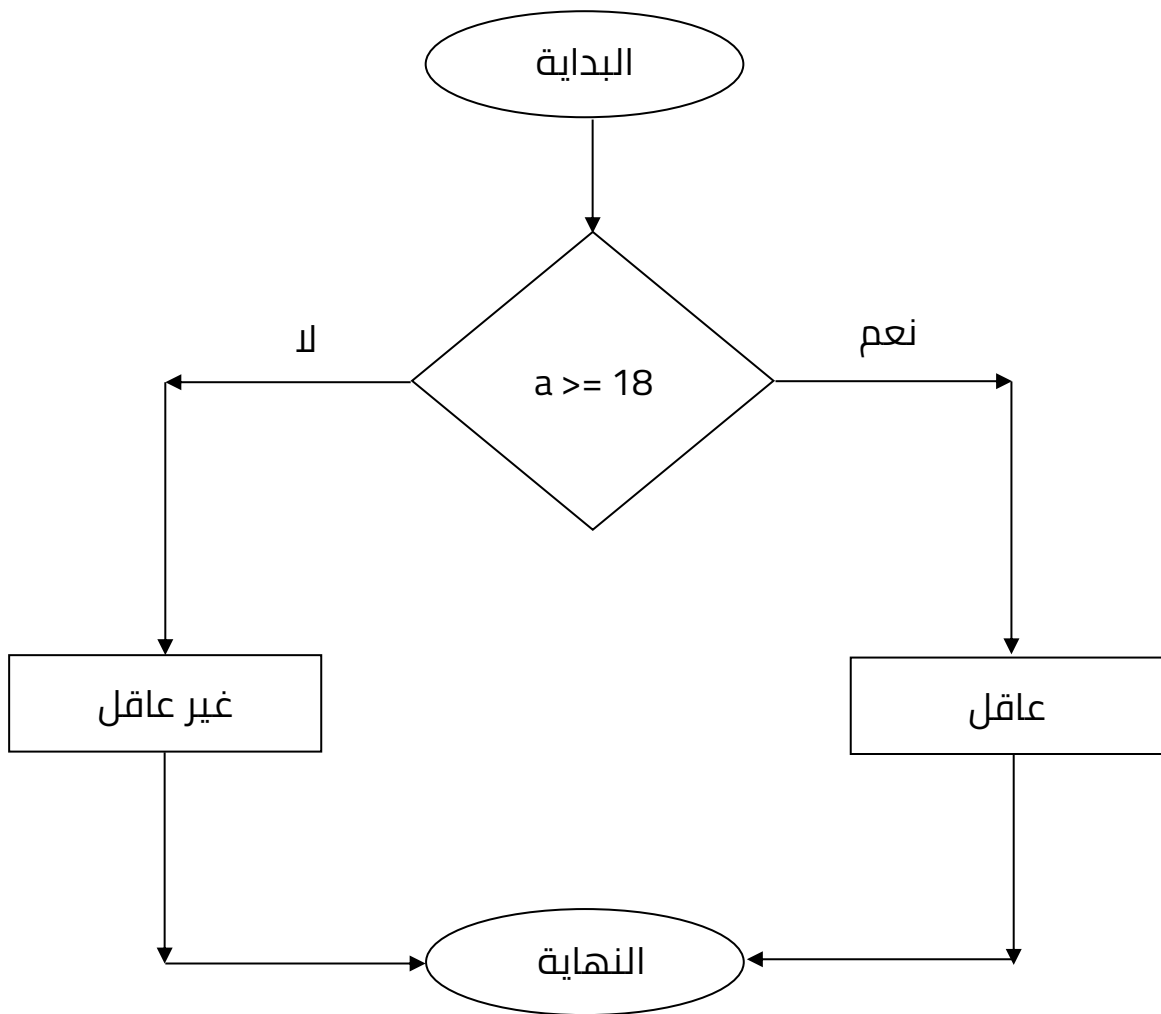
سنحصل على النتيجة التالية عند التشغيل

إذا كان الراتب أقل أو يساوي ٣٠٠٠
تستحق الدعم

إذا كان الراتب أكثر من ٣٠٠٠
لن يتم طباعة أي جملة

❖ جملة else if :

اكتب برنامج يقوم المستخدم بإدخال العمر اذا كان العمر اكبر من او يساوي ١٨ يطبع (عاقل) واذا كان غير ذلك يطبع (غير عاقل) :



- نلاحظ في جملة else if اذا كان العمر أكبر من أو يساوي ١٨ يتم طباعة جملة (عاقل) , واذا كان العمر أصغر من ١٨ يتم طباعة جملة (غير عاقل) .



الكود العام

```
package Test;

import javax.swing.JOptionPane ;

public class Test {

    public static void main(String[] args) {

        String a ;
        a = JOptionPane.showInputDialog(null, "ادخل العمر");

        int b ;
        b=Integer.parseInt(a) ;

        if (b >= 18)

        {
            JOptionPane.showMessageDialog(null, "عاقل");
        }

        Else

            JOptionPane.showMessageDialog(null, "غير عاقل");

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

إذا كان العمر أكبر من أو يساوي ١٨
عاقل

إذا كان العمر أقل من ١٨
غير عاقل

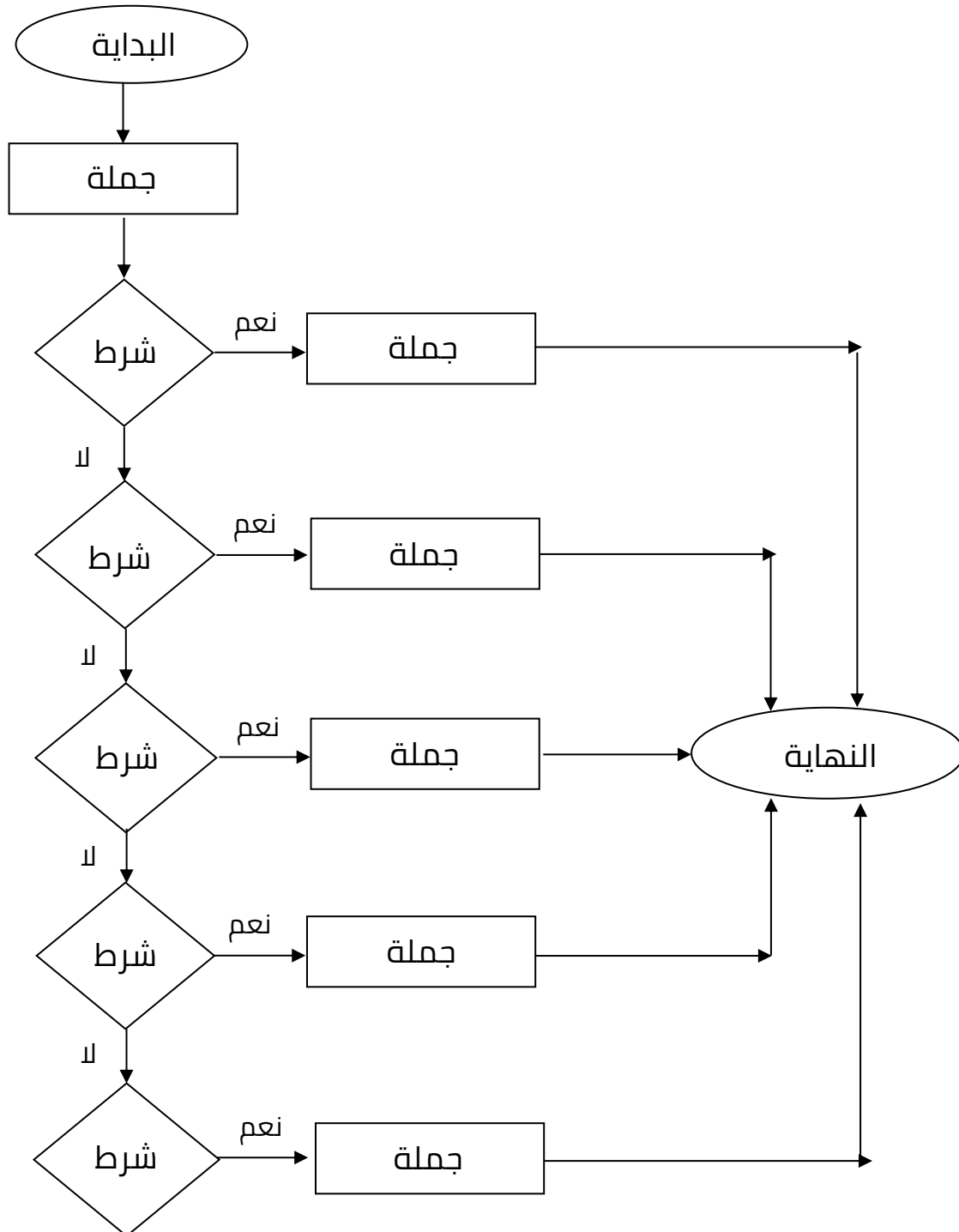
الوحدة السابعة

كتابة جملة IF else , جملة Switch

نظري

ثالثاً : جملة else if

- تستخدم اذا كان هناك اكثر من شرطين (احتماليين) .
- جملة ال else if يتم وضعها في الوسط , أي بين الجملتين if و else .





❖ الشكل العام لجملة else if :

مثال

```
if ( condition )
{
    // إذا كان الشرط صحيحاً نفذ هذا الكود
}

else if ( condition )
{
    // إذا كان الشرط صحيحاً نفذ هذا الكود
}

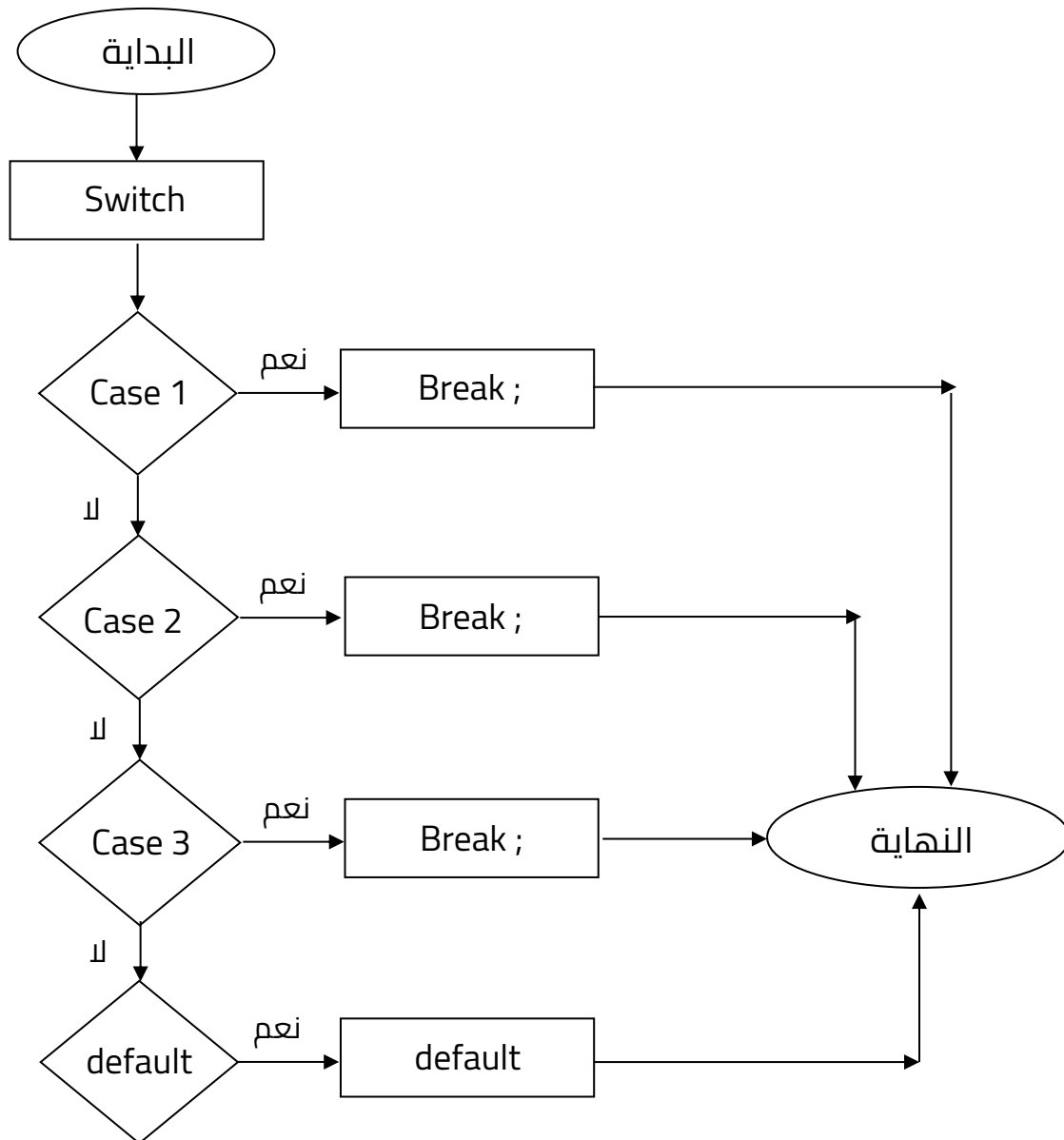
else if ( condition )
{
    // إذا كان الشرط صحيحاً نفذ هذا الكود
}

else
{
    // نفذ هذا الكود في حال لم يتم التعرف على الكود في أي شرط
}
```

- نلاحظ أن جملة **else if** تأتي بين if و else .
- إذا كان الشرط الأول لم يتحقق يذهب مباشرة للشرط الثاني .. وهكذا .
- في حال تحقق أي شرط يتم تنفيذ أمر الطباعة وسيتم تجاوز جميع جمل الشرط التي أتت بعده .
- في الأخير جملة **else** تعني في حال لم يتم تنفيذ أي شرط موجودة في الجمل التي قبلها يتم طباعة أمر معين ومن بعدها يتم إنهاء البرنامج .

رابعاً : جملة Switch

- **switch** تتكون من ثلاث أجزاء مهمة الجزء (switch) وبينه نكتب المتغير
- **case** وهي تماماً كالشرط الذي يوجد بداخل if
- **break** وبهذا الأمر نقوم بمعالجة كل حالة لوحدها
- **default** وهو مقابل else وينفذ عند عدم تحقق أي شرط من الشروط



❖ الشكل العام لجملة Switch :

مثال

```
switch(expression) {
    case value:
        //Statements
        break;
    case value:
        //Statements
        break;
    default:
        //Statements
        break;
}
```

- **switch** تعني إختبر قيمة المتغير الموضوع بين قوسين .
- **expression** هنا يقصد بها المتغير الذي نريد إختبار قيمته .
- نوع المتغير الذي يسمح لنا بإختباره: int - byte - short - char - String - enum .
- **case** تعني حالة, **value** تعني قيمة, و **Statements** تعني أوامر.

و يقصد من هذا كله, أنه في حال كانت قيمة الـ expression تساوي هذه القيمة سيقوم بتنفيذ الأوامر الموضوعة بعد النقطتين :

الآن بعد تنفيذ جميع الأوامر الموضوعة بعد النقطتين , يجب وضع break لكي يخرج من الجملة switch مباشرةً بدل أن ينتقل للـ case التالية الموجودة في الجملة switch .

نستطيع وضع العدد الذي نريده من الـ case بداخل الجملة switch .

تنبيه : الـ expression و الـ value يجب أن يكونا من نفس النوع .

- **default** تعني إفتراضياً و هي نفس فكرة الجملة else , و يمكننا أن لا نضعها أيضاً .

الوحدة السابعة

كتابة جملة IF else , جملة Switch

عملي

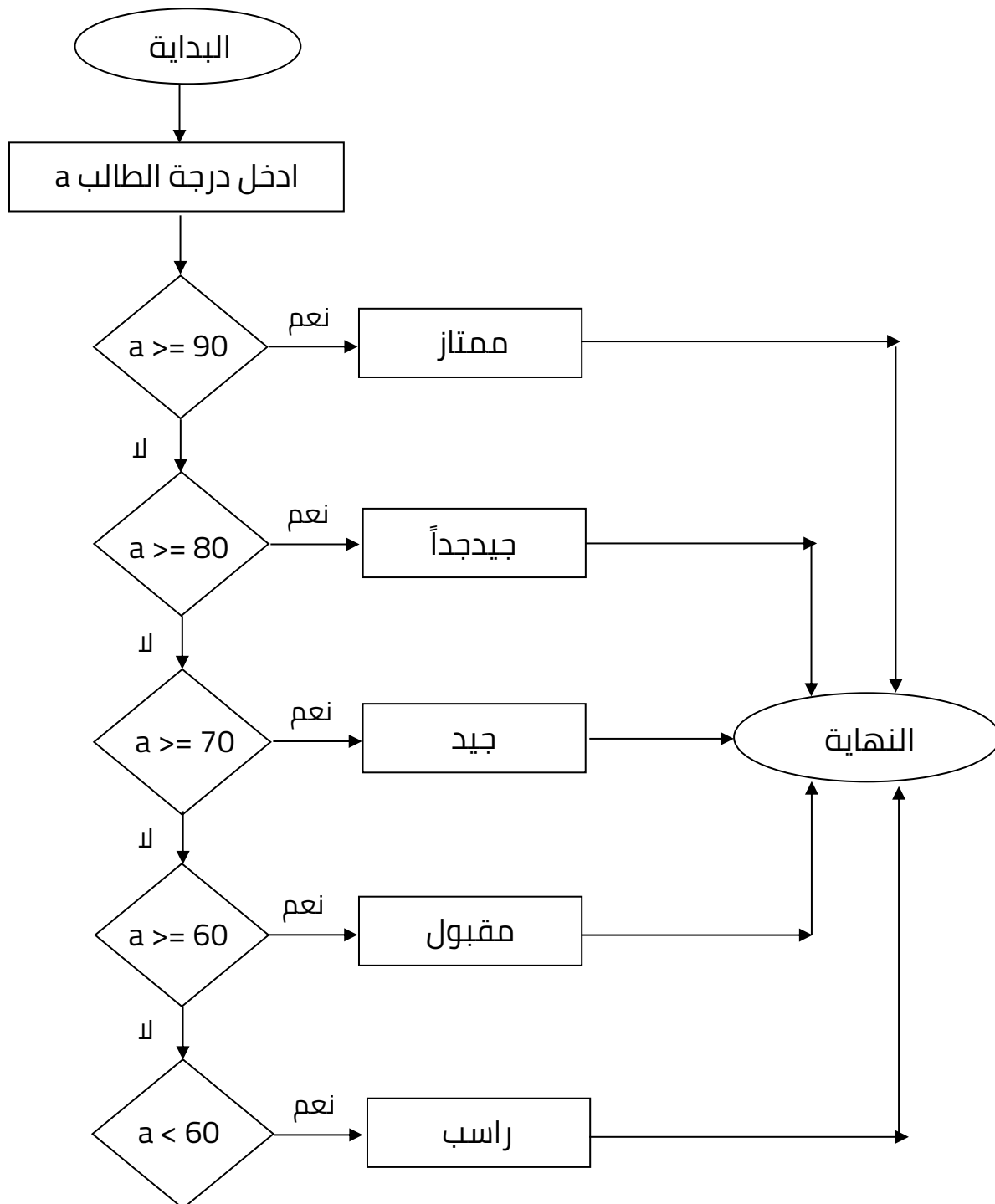


❖ جملة if else :

اكتب برنامج يقوم المستخدم بإدخال الدرجة , إذا كانت الدرجة ٩٠ وأعلى يطبع "ممتاز"

إذا كانت الدرجة ٨٠ وأعلى يطبع "جيد جداً" , إذا كانت الدرجة ٧٠ وأعلى يطبع "جيد"

إذا كانت الدرجة ٦٠ وأعلى يطبع "مقبول" , غير ذلك يطبع "راسب" :





الكود العام

```
package Test;

public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int a = 95 ;

        if (a >= 90)

            System.out.println ("ممتاز") ;

        else if (a >= 80)

            System.out.println ("جيد جداً") ;

        else if (a >= 70)

            System.out.println ("جيد") ;

        else if (a >= 60)

            System.out.println ("مقبول") ;

        else

            System.out.println ("راسب") ;

    }

}
```

سنحصل على النتيجة التالية عند التشغيل

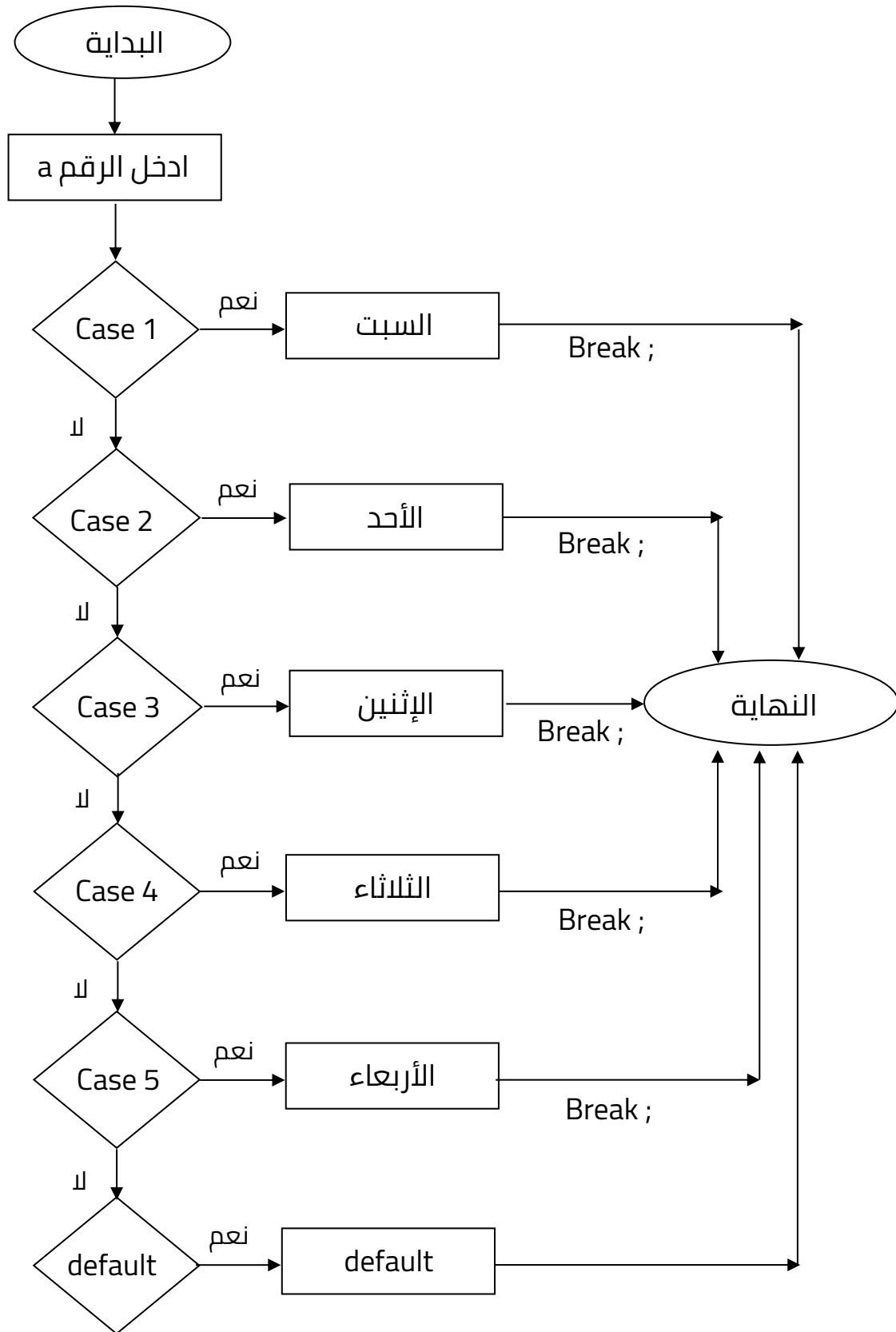
ممتاز

❖ جملة switch :

قم بكتابة برنامج يقوم المستخدم بإدخال الاعداد وسيتم تحويلها البرنامج الى أيام كما في الجدول التالي :

اليوم	الرقم
السبت	١
الاحد	٢
الاثنين	٣
الثلاثاء	٤
الاربعاء	٥
الخميس	٦
الجمعة	٧
لا يوجد يوم بهذا الرقم	غير ذلك

- سنستخدم التالي :
- ✓ **switch** تعني إختبر قيمة المتغير الموضوع بين قوسين .
- ✓ **case** تعني حالة .
- ✓ **break** لكي يخرج من الجملة switch مباشرةً .
- ✓ **default** تعني إفتراضياً .





الكود العام

```
package Test;
public class Test {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int a = 6 ;
        switch (a)
        {
            case 1 :
                System.out.println ("السبت") ;
                break ;
            case 2 :
                System.out.println ("الأحد") ;
                break ;
            case 3 :
                System.out.println ("الاثنين") ;
                break ;
            case 4 :
                System.out.println ("الثلاثاء") ;
                break ;
            case 5 :
                System.out.println ("الأربعاء") ;
                break ;
            case 6 :
                System.out.println ("الخميس") ;
                break ;
            case 7 :
                System.out.println ("الجمعة") ;
                break ;
            default :
                System.out.println ("الرقم بهذا يوم يوجد لا") ;
        }
    }
}
```

سنحصل على النتيجة التالية عند التشغيل

الخميس

الوحدة الثامنة

كتابة جملة `while` , `do while` , `for` حلقة

نظري



ثانياً : جمل الدوران في الجافا

تستخدم لتكرار كتلة من التعليمات البرمجية حتى يتم استيفاء شرط معين , فبدلاً من تكرار كتابة الشفرة ألف مرة مثلاً ستكتبها مرة واحدة و تؤدي لك الحلقات التكرارية بقية المهمة من أجلك بعد أن تحدد لها بعض المتغيرات مثل عدد التكرارات أو متى تبدأ و متى تتوقف , وعادة ما تسمى هذه الوسائل بالحلقات التكرارية .

الحلقات التكرارية من الأدوات المفيدة لكل مبرمج ليؤدي مهامه , ولا غنى عنها أبداً بل وربما تعتبر من أهم الأجزاء في الشفرة التي قد يستخدمها لإزاحة أحمال ثقيلة و رفع سرعته الإنتاجية .

١- جمل الدوران Loop		
جملة for	جملة do-while	جملة while
٢- جمل التحكم في الحلقات		
Continue		

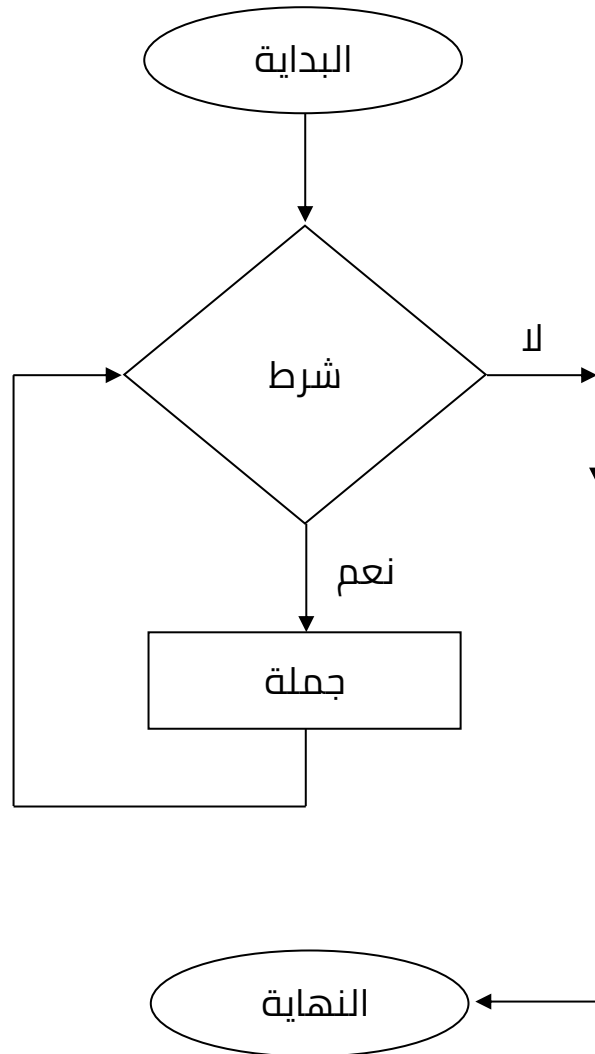


أولاً : جملة While

نستخدم الحلقة while إذا كنا نريد تنفيذ الكود عدة مرات , ولكننا لا نعرف كم مرة بالتحديد لأننا نريد إيقاف التنفيذ إذا تحقق شرط معين .

هذه الحلقة تتوقف عن تكرار نفسها إذا تحقق الشرط الذي وضعناه لها , كأننا نقول : " طالما أن الشرط لم يتحقق إستمر في تكرار الكود " .

❖ طريقة عمل جملة While :



❖ الشكل العام لجملة While :

مثال

```
initialisation;

while( condition )
{
    // statements
    increment أو decrement ;
}
```

initialisation : هي أول خطوة تنفذ في الحلقة و هي تنفذ مرة واحدة فقط على عكس جميع العناصر الموجودة في الحلقة .

في هذه الخطوة نقوم بتعريف متغير (يسمى عداد) .

condition : هي ثاني خطوة تنفذ في الحلقة و هي تنفذ في كل دورة .

في هذه الخطوة نقوم بوضع شرط يحدد متى تتوقف الحلقة, في كل دورة يتم التأكد أولاً إذا تحقق هذا الشرط أم لا , هنا طالما أن نتيجة الشرط تساوي true سيعيد تكرار الكود .

statements : هي الخطوة الثالثة, و تعني تنفيذ جميع الأوامر الموجودة في الحلقة و هي تنفذ في كل دورة.

increment أو decrement : هي الخطوة الرابعة و الأخيرة, و هي تنفذ في كل دورة .

هنا نحدد كيف تزداد أو تنقص قيمة العداد .

تذكر فقط أن جميع هذه الخطوات تتكرر في كل دورة ما عدا أول خطوة, و السبب أننا لا نحتاج إلى تعريف عداد جديد في كل دورة, بل نستعمل العداد القديم و الذي من خلاله نعرف في أي دورة أصبحنا.

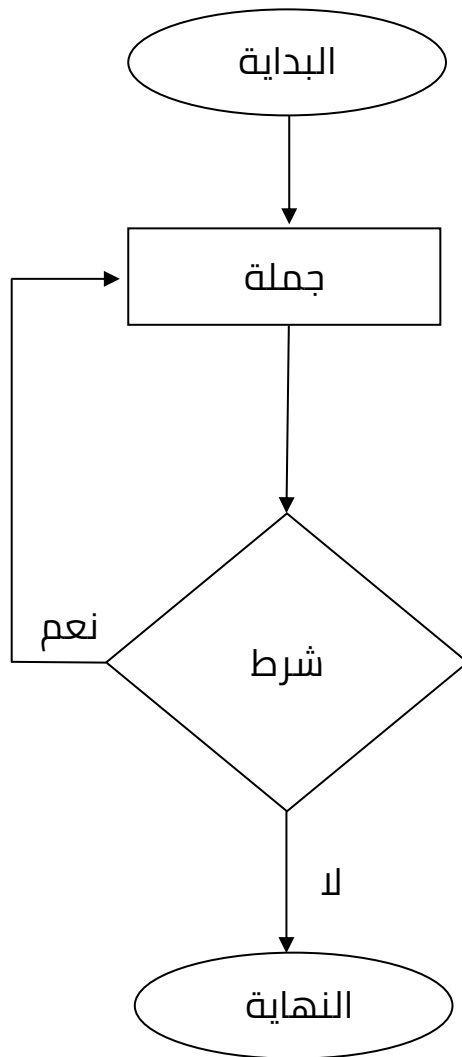
ثانياً : جملة while do

نستخدم الحلقة do while إذا كنا نريد تنفيذ الكود عدة مرات , ولكننا لا نعرف كم مرة بالتحديد لأننا نريد إيقاف التنفيذ إذا تحقق شرط معين .

هذه الحلقة تتوقف عن تكرار نفسها إذا تحقق الشرط الذي وضعناه لها .

الفرق الوحيد بينها وبين الحلقة while أنها تنفذ مرة واحدة على الأقل لأنها تتأكد من الشرط بعد تنفيذ الأوامر وليس قبلهم .

❖ طريقة عمل جملة do While :



❖ الشكل العام لجملة do While :

مثال

```
initialisation;  
  
do{  
    // statements  
    increment أو decrement;  
}  
While( condition );
```

Initialization : هي أول خطوة تنفذ في الحلقة و هي تنفذ مرة واحدة فقط على عكس جميع العناصر الموجودة في الحلقة , في هذه الخطوة نقوم بتعريف متغير (يسمى عداد) .

statements : هي الخطوة الثانية, و تعني تنفيذ جميع الأوامر الموجودة في الحلقة و هي تنفذ في كل دورة .

increment gí decrement : هي الخطوة الثالثة, و هي تنفذ في كل دورة .

هنا نحدد كيف تزداد أو تنقص قيمة العداد .

condition : هي الخطوة الرابعة و الأخيرة و هي تنفذ في كل دورة .

في هذه الخطوة نقوم بوضع شرط يحدد متى تتوقف الحلقة , في نهاية كل دورة يتم التأكد إذا تحقق الشرط أم لا .

هنا طالما أن نتيجة الشرط تساوي true سيعيد تكرار الكود .

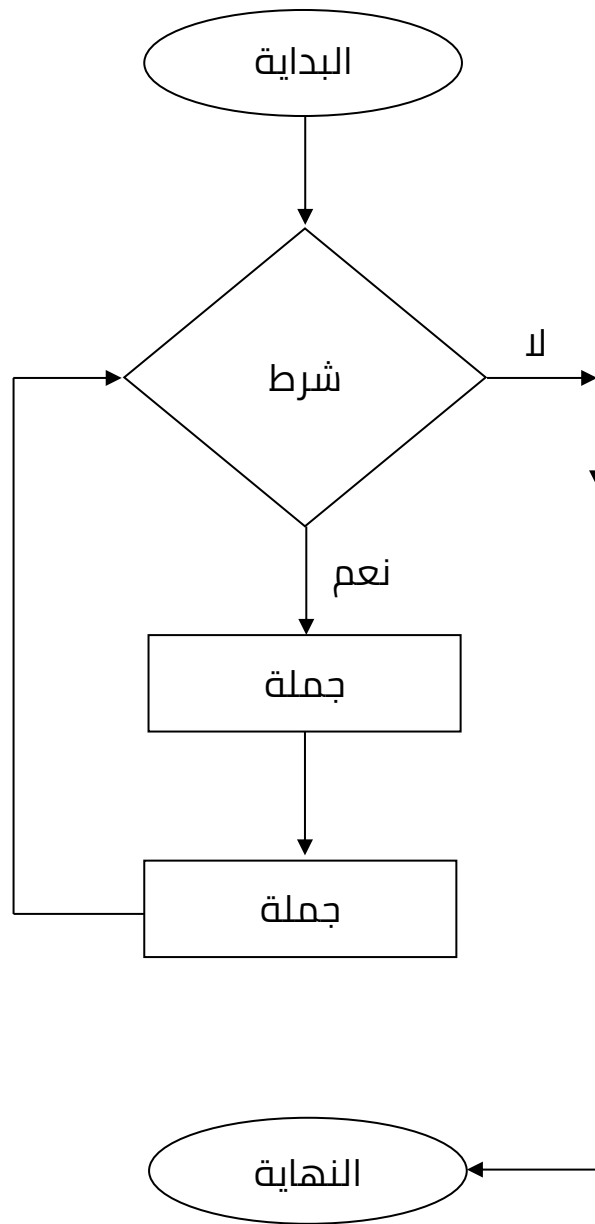
تذكر فقط أن جميع هذه الخطوات تتكرر في كل دورة ما عدا أول خطوة, و السبب أننا لا نحتاج إلى تعريف عداد جديد في كل دورة, بل نستعمل العداد القديم و الذي من خلاله نعرف في أي دورة أصبحنا .



ثالثاً : حلقة for

نستخدم الحلقة for إذا كنا نريد تنفيذ الكود عدة مرات محددة , فمثلاً إذا كنا نريد تنفيذ كود معين ١٠ مرات , نضعه بداخل حلقة تعيد نفسها ١٠ دورات .

❖ طريقة عمل حلقة for :





❖ الشكل العام لحلقة for :

مثال

```
for( initialization ; condition ; increment أو decrement )  
{  
    // statements  
}
```

Initialization : هي أول خطوة تنفذ في الحلقة و هي تنفذ مرة واحدة فقط على عكس جميع العناصر الموجودة في الحلقة. في هذه الخطوة نقوم بتعريف متغير (يسمى عداد) .

condition : هي ثاني خطوة تنفذ في الحلقة و هي تنفذ في كل دورة .

في هذه الخطوة نقوم بوضع شرط يحدد متى تتوقف الحلقة, في كل دورة يتم التأكد أولاً إذا تحقق هذا الشرط أم لا, و نضع بعده ؛ طالما أن نتيجة الشرط تساوي true سيعيد تكرار الكود .

Statements : هي الخطوة الثالثة, و تعني تنفيذ جميع الأوامر الموجودة في الحلقة .

بعد أن تنفذ جميع الأوامر سيمرر إلى الخطوة الأخيرة التي تحدث في نهاية كل دورة و هي إما زيادة قيمة العداد أو إنقاصها .

increment أو decrement : هي الخطوة الرابعة و الأخيرة, و هي تنفذ في كل دورة .

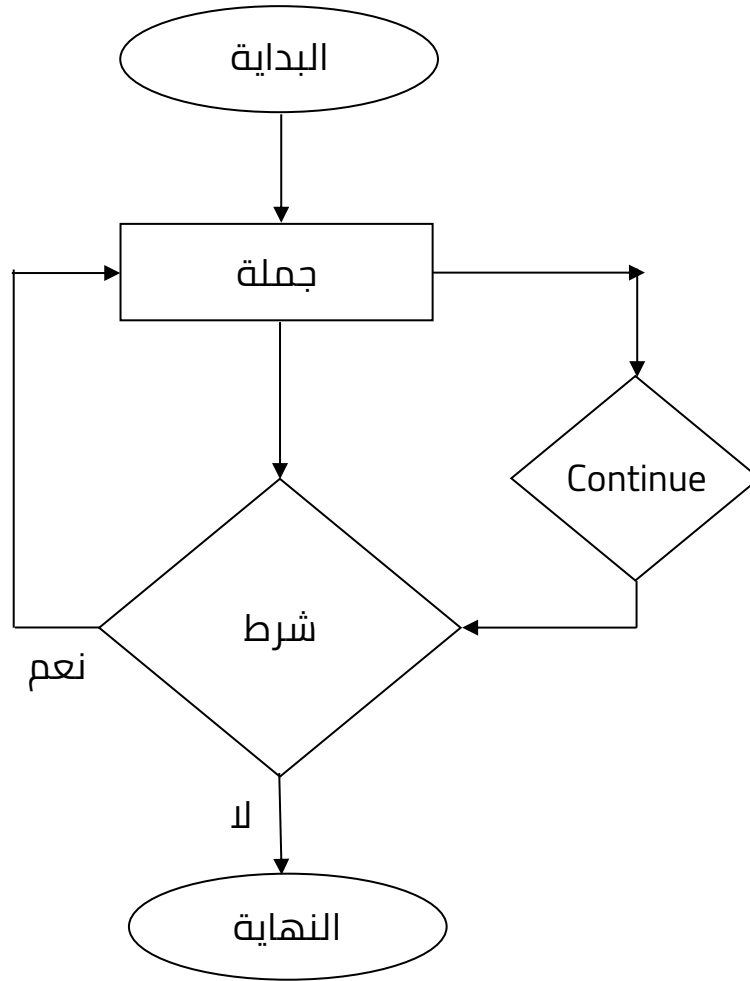
هنا نحدد كيف تزداد أو تنقص قيمة العداد, و لا نضع بعده ؛

تذكر فقط أن جميع هذه الخطوات تتكرر في كل دورة ما عدا أول خطوة, و السبب أننا لا نحتاج إلى تعريف عداد جديد في كل دورة, بل نستعمل العداد القديم و الذي من خلاله نعرف في أي دورة أصبحنا.

رابعاً : جملة التحكم Continue

نستخدم الجملة continue لتجاوز تنفيذ كود معين في الحلقة .

❖ طريقة عمل جملة التحكم Continue :



❖ طريقة تعريف جملة التحكم Continue :

مثال

```
Continue ;
```

تتألف هذه الجملة من أمر واحد و يكتب على سطر منفرد .

الوحدة الثامنة

كتابة جملة while , do while , for حلقة

عملي



❖ جملة While :

قم بكتابة برنامج يقوم بطباعة الاعداد التاليه من ١ الى ٥ باستخدام جملة **while** :

٥	٤	٣	٢	١
---	---	---	---	---

الكود العام

```
public class Main {  
    public static void main(String[] args) {  
        // هنا قمنا بتعريف المتغير الذي استخدمناه كعداد في الحلقة  
        int i=1;  
        // تظل تنفذ الأوامر الموضوعة فيها طالما أن قيمة العدد لا تزال  
        // هنا أنشأنا حلقة while أصغر أو تساوي ٥  
        while( i<=5 )  
        {  
            // في كل دورة سيتم طباعة قيمة العداد ثم إضافة ١ عليها  
            System.out.println( i );  
            i++;  
        }  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

1 2 3 4 5



❖ جملة do While :

قم بكتابة برنامج يقوم بطباعة الاعداد التاليه من ١ الى 8 باستخدام جملة **do while** :

٨	٧	٦	٥	٤	٣	٢	١
---	---	---	---	---	---	---	---

الكود العام

```
public class Main {  
    public static void main(String[] args) {  
        // هنا قمنا بتعريف المتغير الذي استخدمناه كعداد في الحلقة  
        int i=1;  
  
        // تظل تنفذ الأوامر الموضوعة فيها طالما أن قيمة العدد لا تزال  
        // هنا أنشأنا حلقة while أصغر أو تساوي ٨  
  
        do  
        {  
            // في كل دورة سيتم طباعة قيمة العداد ثم إضافة ١ عليها  
            System.out.println( i );  
            i++;  
        }  
  
        while( i<=8 );  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

1 2 3 4 5 6 7 8



❖ الحلقة for :

قم بكتابة برنامج يقوم بطباعة جميع الأرقام من 3 الى 10 باستخدام حلقة for :

١٠	٩	٨	٧	٦	٥	٤	٣
----	---	---	---	---	---	---	---

الكود العام

```
public class Main {  
    public static void main(String[] args) {  
        // تتألف من ٨ دورات . في كل دورة تطبع قيمة العداد المستخدم  
        // هنا قمنا بإنشاء حلقة for فيها  
        for( int i=3; i<=10; i++ )  
        {  
            System.out.println( i );  
        }  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

3 4 5 6 7 8 9 10



❖ جملة التحكم Continue :

قم بكتابة برنامج يقوم بطباعة الاعداد الفردية من ١ الى ١٦ بإستخدام حلقة **for** :

١٠	١٣	١١	٩	٧	٥	٣	١
----	----	----	---	---	---	---	---

الكود العام

```
public class Main {  
    public static void main(String[] args) {  
  
        for( int i=1; i<=16; i++ )  
        {  
            if( i%2 == 0 )  
                continue;  
  
            System.out.println( i );  
        }  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

```
1 3 5 7 9 11 13 15
```

- في المثال السابق قمنا بتعريف حلقة تطبع جميع الأرقام من ١ إلى ١٦ ما عدا الأعداد الفردية إستخدمنا الجملة continue لجعل الحلقة تتجاوز الأعداد الفردية في الحلقة أي لن يتم تنفيذ أمر الطباعة عندما تصبح قيمة العداد (i) عدد فردي .

الوحدة التاسعة

الدوال في لغة الجافا

نظري

مفهوم الدوال :

دالة : تعني Function or Method في اللغة الإنجليزية , و هي عبارة عن مجموعة أوامر مجمعة في مكان واحد و تنفذ عندما نقوم باستدعائها .

كما أن جافا تحتوي على مجموعة كبيرة جداً من الدوال الجاهزة التي يمكنك إستعمالها مباشرة , كما يمكننا من إنشاء دوال خاصة تؤدي وظائف محددة .

❖ مزايا استخدام الدوال :

- ✓ عدم الحاجة إلى تكرار التعليمات داخل البرنامج حيث يتم إنشاء الدالة مرة واحدة ويمكن استدعائها أكثر من مرة عند الحاجة إليها .
- ✓ باستخدام الدوال يصبح البرنامج أكثر وضوحاً .
- ✓ باستخدام الدوال الجاهزة يمكن توفير الكثير من الوقت والجهد .

أنواع الدوال :

١- دوال جاهزة يمكن أن توفرها لغة الجافا (Build-in) :

وهي طرق تكون مكتوبة وموجودة مسبقاً ويقوم المبرمج باستخدامها فقط , وتكون موجودة في ما يعرف بالمكتبات Libraries الخاصة بلغة الجافا .

مثل : الدوال الرياضية , دوال التعامل مع النصوص , الدوال العامة .

٢- دوال يمكن تعريفها عن طريق المستخدم (User- defined) :

وهي مجموعة الدوال التي يتم انشاءها من قبل المبرمج لأداء وظيفة معينة .

❖ دوال الطباعة هي من الدوال الجاهزة في لغة جافا .

أمثلة

```
System.out.print();
System.out.println();
System.out.printf();
```

❖ **قابلية الوصول Modifier Access :**

- عامه Public : يمكن الوصول إليها من كافة الفئات في المشروع .
- خاصة Private : لا يمكن الوصول إليها إلا من داخل الفئة المعرفة فيها .
- محمية Protected : لا يمكن الوصول إليها من خلال الفئة المعرفة فيها الفئات الموروثة منها .

❖ **المشاركة بين الكائنات (Static , non static) :**

- الدوال يتم تعريفها داخل الفئات والتي يمكن ان نشق منها مجموعة من الكائنات وفي هذه الحالة توجد هنالك نوعين من الدوال :

• **مشتركة Static (Class member) :**

أي أن هذه الدالة مشتركة (لها موقع واحدة في الذاكرة) بين كافة الكائنات المشتقة من الفئة المحتوية على الدالة وعند استدعاء هذا النوع من الدوال لانتاج الى اشتقاق كائن من الفئة المحتوية على الدالة.

• **غير مشتركة Non Static (instance member) :** اي انه لكل كائن مشتق من الفئة

قيما خاصة لكافة متغيرات الدالة وفي مواقع مختلفة من الذاكره ولاستدعاء هذه الدالة يجب اولاً اشتقاق كائن (object) من الفئة المحتوية على الدالة .

❖ القيمة المرجعة لسطر الاستدعاء : Returned Value to calling code :

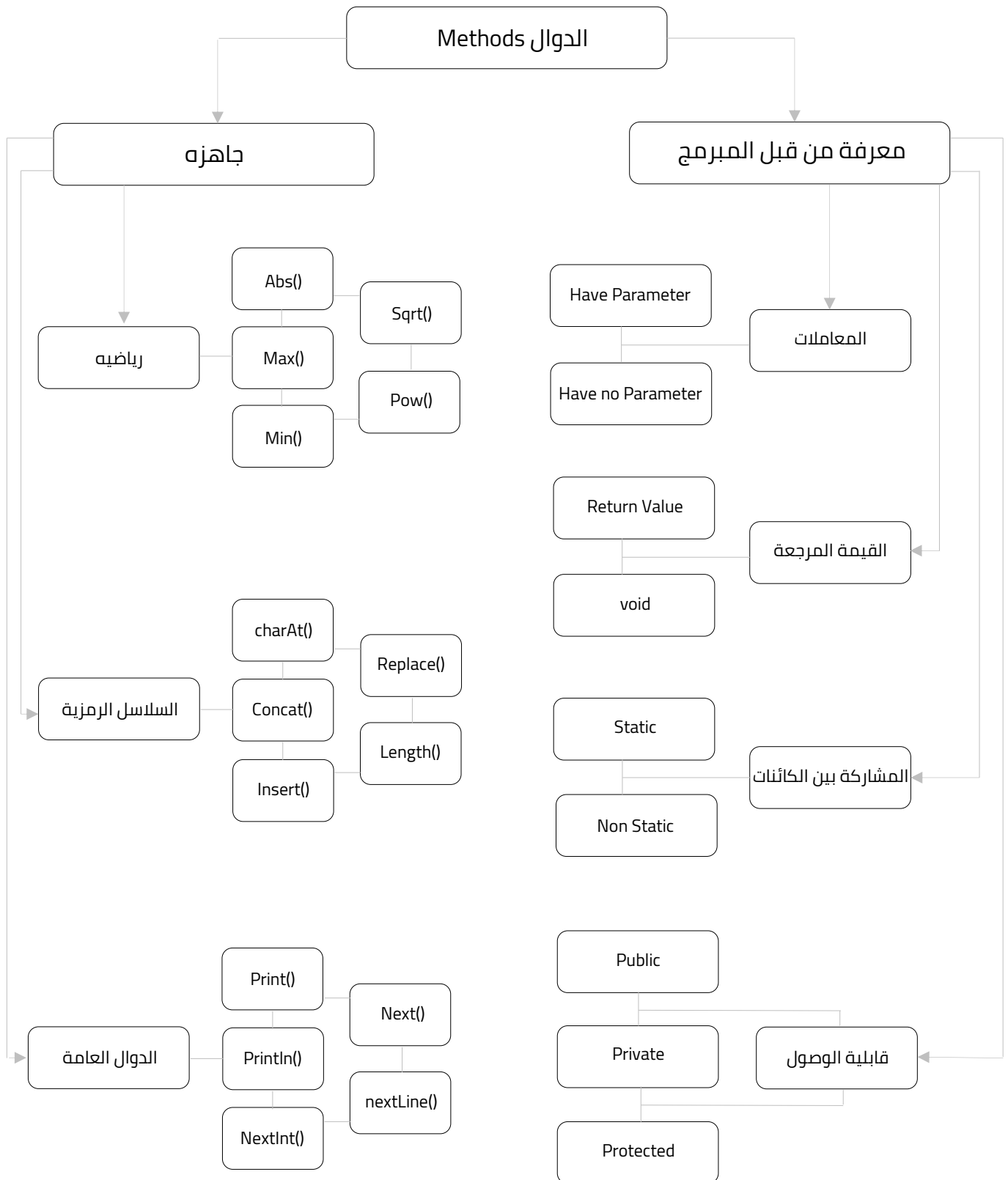
- دوال ترجع قيمة (Return Value) (Getter) : هذا النوع من الدوال يقوم بتنفيذ تعليمات محددة ويقوم بإرجاع قيمة (يتم تحديد نوعها أثناء تعريف الدالة) الى سطر الاستدعاء بعد انتهاء التنفيذ ويجب أن يحتوي جسم الدالة على الكلمة المحبوزة return .
- دوال لا ترجع قيمة (void) (Return no Value) (Setter) : هذا النوع من الدوال تقوم بتنفيذ تعليمات محددة دون أن تقوم بإرجاع قيمة الى سطر الاستدعاء (Calling code) بعد انتهاء التنفيذ .
- سطر الاستدعاء Calling Code : هو السطر الذي تم عنده استدعاء الدالة في الدالة الرئيسية (Main Method) .

❖ احتوائها على المعاملات With or without Parameters :

- دوال لا تحتاج الى تمرير معاملات Have no Parameter : وهي دوال لا تحتاج الى تمرير قيم أثناء استدعائها حيث لا يتم كتابة اي قيم بين قوسي الدالة .
- دوال تحتاج تمرير الى معاملات Have Parameter : وهي دوال تحتاج الى تمرير (ارسال) معاملات (ثوابت او متغيرات) أثناء استدعائها .
- يتم كتابة المعاملات كقيم ثابتة او متغيرات تحمل قيما بين قوسين امام اسم الدالة أثناء استدعاء الدالة .

▪ **المعاملات** هي عبارة عن قيم (متغيرات او ثوابت) يتم تمريرها إلى الدالة أثناء استدعائها من خلال كتابة قيم او متغيرات مناظرة للمتغيرات المعرفة في راس الدالة في جملة الاستدعاء .

مخطط يوضح تصنيف الدوال في لغة الجافا :



بناء الدوال :

❖ عند تعريف أي دالة في الجافا عليك إتباع الشكل التالي :

أمثلة

```
modifier returnType methodName (Parameters List) {  
  
    //Method Body  
}
```

▪ **modifier** : يحدد طريقة الوصول للدالة .

▪ **returnType** : يحدد النوع الذي سترجعه الدالة عندما تنتهي أو إذا كانت لا ترجع أي قيمة .

▪ **methodName** : يمثل الإسم الذي نعطيه للدالة و الذي من خلاله يمكننا استدعاءها

▪ **Parameters List** : المقصود بها الباراميترات (وضع الباراميترات إختياري) .

▪ **Method Body** : تعني جسم الدالة, و المقصود بها الأوامر التي نضعها في الدالة .

▪ **returnType** في الدالة يمكن أن يكون أي نوع من أنواع البيانات الموجودة في جافا (int , double , Boolean , String إلخ..) .

و يمكن وضع إسم لكلاس معين , و هنا يكون القصد أن الدالة ترجع كائن من هذا الكلاس

في حال كانت الدالة لا ترجع أي قيمة , يجب وضع الكلمة void مكان الكلمة **returnType** .

الوحدة التاسعة

الدوال في لغة الجافا

عملي



❖ في المثال التالي قمنا بتعريف دالة إسمها **welcomeMessage** , نوعها **void** و تحتوي على أمر طباعة فقط , بعدها قمنا باستدائها في الدالة **main()** حتى يتم تنفيذ أمر الطباعة الموضوعه فيها .

الكود العام

```
public class Main {  
    // هنا قمنا بتعريف دالة إسمها welcomeMessage. عند إستدعاءها تطبع جملة للترحيب  
    public static void welcomeMessage() {  
        System.out.println("Hello World");  
    }  
  
    public static void main(String[] args) {  
        // هنا قمنا باستدعاء الدالة welcomeMessage لطباعة جملة الترحيب الموضوعه فيها  
        welcomeMessage();  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

```
Hello World
```



❖ في المثال التالي قمنا بتعريف دالة إسمها **sum** , عند إستدعاءها نعطيها عددين فترجع ناتج جمع هذين العددين , بعدها قمنا باستدائها في الدالة **main()** .

الكود العام

```
public class Main {  
    // هنا قمنا بتعريف دالة إسمها sum عند إستدعاءها نعطيها عددين فترجع ناتج جمع هذين العددين  
    public static int sum(int a, int b) {  
        return a+b ;  
    }  
  
    public static void main(String[] args) {  
        // هنا قمنا باستدعاء الدالة sum لحساب ناتج جمع العددين ٥ و ١٠  
        System.out.println( "10 + 5 = " + sum(10, 5) ) ;  
    }  
}
```

سنحصل على النتيجة التالية عند التشغيل

```
10 + 5 = 15
```

الوحدة العاشرة

المصفوفات ذات البعد الواحد وذات البعدين

نظري



مفهوم المصفوفات :

المصفوفة تسمى Array في اللغة الإنجليزية .
يمكنك تصوّر المصفوفة كمتغير واحد بإمكانه تخزين عدة قيم , إذاً المصفوفة كأنها متغير واحد يحتوي على عدة خانات .

أنواع المصفوفات :

- ❖ **ذات بعد واحد** أي One dimensional array , في مادة الرياضيات تسمى Vector .
- ❖ **ذات بعدين** أي Two dimensional array , في مادة الرياضيات تسمى Matrix .
- ❖ **ذات عدة أبعاد** أي Multidimensional array , في مادة أيضاً الرياضيات تسمى Matrix .

فوائد المصفوفات :

لنفترض أننا نريد إنشاء برنامج يطلب من المستخدم إدخال ١٠٠ عدد صحيح , و بعد إدخال جميع الأعداد نريده أن يعرض له جميع الأعداد التي قام هو بإدخالها .

• للوهلة الأولى ستفكر كالتالي :

- هل أنا بحاجة إلى ١٠٠ متغير لكي أحفظ ١٠٠ قيمة !
 - هل سأطلب من المستخدم ١٠٠ مرة أن يدخل عدداً صحيحاً !
 - إذا أردت أن أعرض له الأعداد التي قام بإدخالها, هل سأكتب دالة الـ **Print()** أيضاً ١٠٠ مرة !
- الجواب هو حتماً كلا , لأنك لو كنت ستفعل ذلك, ستضطر إلى كتابة أكثر من ٤٠٠ سطر , في حين أنك تستطيع تنفيذ البرنامج بـ ٥ أسطر فقط لو استخدمت مصفوفة لتخزين الأشياء التي أدخلها المستخدم .



A1

عبارة عن مصفوفة ذات بعد واحد لأنها تتألف من سطر واحد فقط
هنا يوجد سطر واحد فقط يتألف من 0 أعمدة

A1

10	2	5	13	-4
----	---	---	----	----

A2

عبارة عن مصفوفة ذات بعدين لأنها تتألف من أسطر وأعمدة
هنا يوجد 3 أسطر يتألف من 0 أعمدة

A2

10	2	5	13	-4
6	7	-1	0	3
8	4	9	21	5

A3

عبارة عن مصفوفة متعددة الأبعاد لأنها تتألف من أسطر وأعمدة داخلية
هنا يوجد 3 أسطر تتألف من 3 أعمدة وكل عامود يتألف من 3 أعمدة أخرى

A3

9	5	8	0	47	3	15	60	7
7	3	4	7	9	72	66	2	21
8	79	48	12	-5	25	70	10	4

❖ تستخدم المصفوفات للأسباب التالية :

- ✓ تقليل الوقت و الجهد على المبرمج .
- ✓ السرعة في الأداء .
- ✓ تقليل حجم الكود .
- ✓ إمكانية الوصول للقيم بطريقة سريعة و سهلة جداً .

❖ طرق تعريف المصفوفات :

لتعريف المصفوفات , يوجد طريقتين :

- تعرّف المصفوفة و تعطيها القيم لاحقاً .
- تعرّف المصفوفة و تعطيها القيم مباشرةً عند تعريفها .

❖ طريقة تعريف مصفوفة ذات البعد الواحد وذات البعدين :

- تحديد نوع البيانات (int, float , double , String) .
- نضع الرمز كالتالي :

مصفوفة ذات البعدين matrix	مصفوفة ذات البعد الواحد vector
نضع الرمز [] []	نضع الرمز []

❖ طريقة تعريف المصفوفة بدون إعطائها قيم أولية :

مثال

```
//مصفوفة ذات بعد واحد
int[] vector = new int[5];

//مصفوفة ذات بعدين
int[][] matrix = new int[3][5];
```



شكل المصفوفة في الذاكرة

Vector []

value	0	0	0	0	0
index	0	1	2	3	4
	العنصر الأول	العنصر الثاني	العنصر الثالث	العنصر الرابع	العنصر الخامس

Matrix [][]

index	0	1	2	3	4	
0	0	0	0	0	0	السطر الأول
1	0	0	0	0	0	السطر الثاني
2	0	0	0	0	0	السطر الثالث
	العمود الأول	العمود الثاني	العمود الثالث	العمود الرابع	العمود الخامس	

❖ طريقة إعطاء قيم لعناصر المصفوفة :

مثال

قيم عناصر مصفوفة ذات بعد واحد

vector[0] = 7;

vector[1] = 40;

vector[2] = -20;

قيم عناصر مصفوفة ذات بعدين

matrix[0][0] = 6;

matrix[1][2] = -3;

matrix[2][4] = 100;



❖ طريقة عرض القيم المخزنة في عناصر المصفوفة :

مثال

عرض عناصر مصفوفة ذات بعد واحد

```
System.out.print("vector[0]: " + vector[0] + "\n" );  
System.out.print("vector[1]: " + vector[1] + "\n" );  
System.out.print("vector[2]: " + vector[2] + "\n" );  
System.out.print("vector[3]: " + vector[3] + "\n" );  
System.out.print("vector[4]: " + vector[4] + "\n" );
```

قيم عناصر مصفوفة ذات بعدين

هنا قمنا بعرض جميع قيم العناصر الموجودة في السطر الأول //

```
System.out.print("matrix[0][0]: " + matrix[0][0] + "\n" );  
System.out.print("matrix[0][1]: " + matrix[0][1] + "\n" );  
System.out.print("matrix[0][2]: " + matrix[0][2] + "\n" );  
System.out.print("matrix[0][3]: " + matrix[0][3] + "\n" );  
System.out.print("matrix[0][4]: " + matrix[0][4] + "\n\n" );
```

هنا قمنا بعرض جميع قيم العناصر الموجودة في السطر الثاني //

```
System.out.print("matrix[1][0]: " + matrix[1][0] + "\n" );  
System.out.print("matrix[1][1]: " + matrix[1][1] + "\n" );  
System.out.print("matrix[1][2]: " + matrix[1][2] + "\n" );  
System.out.print("matrix[1][3]: " + matrix[1][3] + "\n" );  
System.out.print("matrix[1][4]: " + matrix[1][4] + "\n\n" );
```

هنا قمنا بعرض جميع قيم العناصر الموجودة في السطر الثالث //

```
System.out.print("matrix[2][0]: " + matrix[2][0] + "\n" );  
System.out.print("matrix[2][1]: " + matrix[2][1] + "\n" );  
System.out.print("matrix[2][2]: " + matrix[2][2] + "\n" );  
System.out.print("matrix[2][3]: " + matrix[2][3] + "\n" );  
System.out.print("matrix[2][4]: " + matrix[2][4] + "\n\n" );
```

..

❖ المبادئ التي عليك اتباعها أو تقليدها مع المصفوفات :

- إستخدم الحلقة **for** في حال أردت الوصول لجميع عناصر المصفوفة و إجعلها تبدأ من

0 إلى عدد عناصرها ناقص 1

- إستخدم الحلقة **while** أو **do while** في حال لم تكن تريد الوصول لجميع العناصر .

إستخدم الحلقة **do while** في حال كنت تريد جعل المستخدم يدخل قيمة تستوفي شرط

معين .

مفهوم الخاصية length :

الخاصية **length** عبارة عن ثابت يمكنك إعتباره متغير عادي تملكه كل مصفوفة يتم تعريفها بشكل تلقائي , تستخدم هذه الخاصية لمعرفة عدد عناصر المصفوفة , أو كما يقال لمعرفة حجم المصفوفة بشكل عام , نحتاج إستخدامها عند بناء كود للتعامل مع المصفوفة مهما كان حجمها .

❖ طريقة إستخدام الخاصية length مع المصفوفات :

لإستخدام الخاصية length الموجودة في أي مصفوفة نضع إسم المصفوفة , ثم نقطة , ثم الكلمة length كالتالي :

```
ArrayName.length ;
```

❖ الفرق بين المتغير والمصفوفة :

المتغير	المصفوفة
يحتوي على قيمة واحدة من نوع واحد	يحتوي على عدة قيم من نوع واحد

الوحدة العاشرة

المصفوفات ذات البعد الواحد وذات البعدين

عملي



التعامل مع المصفوفة ذات البعد الواحد :

- أكتب برنامج يطلب من المستخدم إعطائه عدد يمثل عدد عناصر مصفوفة إسمها **vector** و يخزنه في متغير إسمه **N**.
- ثم يطلب من المستخدم إدخال قيمة لكل عنصر من عناصر المصفوفة **vector**.
- ثم يعرض للمستخدم ناتج جمع جميع قيم عناصر المصفوفة **vector**.

الكود العام

```
import java.util.Scanner;

public class Vector {
    public static void main (String[] args) {

        Scanner input = new Scanner(System.in);

        int N;
        int S = 0;
        int[] vector;

        do
        {
            System.out.print("Enter the length of the vector: ");
            N = input.nextInt();
        }
        while ( N <= 0 );

        vector = new int[N];

        for (int i=0; i <=N-1; i++)
        {
            System.out.print("Enter vector[" +i+ "]: ");
            vector[i] = input.nextInt();
            S = S + vector[i];
        }
        System.out.print("The sum of all elements is: " +S+ "\n" );
    }
}
```



سنحصل على النتيجة التالية عند التشغيل

```
Enter the length of the vector: 3
Enter vector[0]: 2
Enter vector[1]: 5
Enter vector[2]: 7
The sum of all elements is: 14
```

❖ شرح الكود :

```
int N;
int S = 0;
int[] vector;
```

- هنا قمنا بتجهيز المتغير N لتخزين عدد عناصر المصفوفة، و هو أول شيء سيطلب من المستخدم إدخاله .
- و المتغير S لتخزين مجموع قيم عناصر المصفوفة.
- و المصفوفة vector و التي لم يتم تحديد عدد عناصرها.

```
do
{
    System.out.print("Enter the length of the vector: ");
    N = input.nextInt();
}
while ( N <= 0 );
```


هنا سيطلب من المستخدم إدخال عدد عناصر المصفوفة، بعدها سيتم تخزينه في المتغير N . بعدها سيتم فحص قيمة المتغير N . إذا كانت أصغر أو تساوي ٠، سيطلب من المستخدم إدخال العدد من جديد، إذاً هذه الحلقة تضمن أن لا يقوم المستخدم بإدخال عدد أصغر أو يساوي ٠.

```
vector = new int[N];
```

هنا سيتم تحديد عدد عناصر المصفوفة و الذي يساوي قيمة العدد N بعدها سيتم طباعة عدد عناصر المصفوفة باستخدام الخاصية `length`. لو وضعنا N بدل `vector.length` لكان الجواب نفسه لأن عدد العناصر أيضاً يساوي قيمة المتغير N .

```
for (int i=0; i <=N-1; i++)
{
    System.out.print("Enter vector[" +i+ "]: ");
    vector[i] = input.nextInt();
    S = S + vector[i];
}
System.out.print("The sum of all elements is: " +S+ "\n");
```

هنا أنشأنا حلقة تبدأ من `index` العنصر الأول في المصفوفة إلى آخر عنصر موجود فيها. في كل دورة من دورات الحلقة i سيحدث التالي: سيطلب من المستخدم إدخال قيمة لعنصر محدد من عناصر المصفوفة. بعدها سيتم تخزين العدد الذي سيدخله في هذا العنصر. بعدها سيتم إضافة قيمة العنصر على قيمة المتغير S . في الأخير سيتم عرض ناتج جمع جميع عناصر المصفوفة `vector` المخزن في المتغير S .



التعامل مع المصفوفة ذات البعدين :

- أكتب برنامج يعرّف مصفوفة إسمها matrix تتألف من ٣ أسطر و ٣ أعمدة.
- ثم يطلب من المستخدم إدخال قيم لها.
- ثم يعرض للمستخدم ناتج جمع جميع قيم عناصرها.

الكود العام

```
import java.util.Scanner;

public class Matrix {

    public static void main (String[] args) {

        Scanner input = new Scanner(System.in);

        int[][] matrix = new int[3][3];
        int S = 0;

        for (int i=0; i<3; i++)
        {
            for (int j=0; j<3; j++)
            {
                System.out.print("Enter matrix["+i+"]["+j+"]: ");
                matrix[i][j] = input.nextInt();
            }
            System.out.print("\n");
        }

        for (int i=0; i<3; i++)
        {
            for (int j=0; j<3; j++)
            {
                S = S + matrix[i][j];
            }
        }

        System.out.print("The sum of all elements is: " +S+ "\n");

    }
}
```



سنحصل على النتيجة التالية عند التشغيل

```
Enter matrix[0][0]: 2
Enter matrix[0][1]: 4
Enter matrix[0][2]: 6

Enter matrix[1][0]: 1
Enter matrix[1][1]: 2
Enter matrix[1][2]: 3

Enter matrix[2][0]: 2
Enter matrix[2][1]: 2
Enter matrix[2][2]: 4

The sum of all elements is: 26
```

❖ شرح الكود :

```
int[][] matrix = new int[3][3];
int S = 0;
```

هنا قمنا بتجهيز المصفوفة matrix و حددنا أنها تتألف من ٣ أسطر و ٣ أعمدة .
و المتغير S الذي سنستخدمه لاحقاً لتخزين ناتج جمع قيم جميع العناصر الموجودة في
المصفوفة لذلك أعطيناه القيمة ٠ كقيمة أولية .

```
for (int i=0; i<3; i++)
{
    for (int j=0; j<3; j++)
    {
        System.out.print("Enter matrix["+i+"]["+j+"]: ");
        matrix[i][j] = input.nextInt();
    }
    System.out.print("\n");
}
```



هنا قمنا بإنشاء الحلقتين *i* و *z* لجعل المستخدم يدخل قيمة لكل عنصر في المصفوفة. الحلقة *i* للانتقال من سطر إلى آخر في المصفوفة، و الحلقة *z* للانتقال من عمود إلى آخر في كل سطر في المصفوفة.

هنا في كل دورة من دورات الحلقة *i* سيتم إنشاء حلقة *z* للمرور على جميع العناصر الموجودة في نفس السطر.

في كل دورة من دورات الحلقة *z* سيطلب من المستخدم إدخال قيمة لعنصر، ثم سيتم تخزينها في هذا العنصر.

بعد توقف الحلقة *z*، أي بعد إعطاء قيم لجميع العناصر الموجودة في نفس السطر، سيتم النزول على سطر جديد بسبب الرمز `\n`.

```
for (int i=0; i<3; i++)  
{  
    for (int j=0; j<3; j++)  
    {  
        S = S + matrix[i][j];  
    }  
}
```

هنا قمنا بإنشاء الحلقتين *i* و *z* للوصول إلى جميع قيم عناصر المصفوفة.

كل عنصر يتم الوصول إليه، يتم إضافة قيمته على قيمة المتغير *S*.

إذاً عند توقف الحلقتين *i* و *z* سيكون المتغير *S* يحتوي على ناتج جميع قيم عناصر المصفوفة.

```
System.out.print("The sum of all elements is: " +S+ "\n");
```

هنا قمنا بعرض قيمة المتغير *S* كناتج جمع جميع قيم العناصر الموجودة في المصفوفة.

الوحدة الحادي عشر

الكائنات والكلاسات Classes and Objects

نظري

تعريف الكلاس Classes :

Class : نكتبها كلاس في العربية . و الكلاس عبارة عن حاوية كبيرة تستطيع أن تحتوي على كل الكود من متغيرات و دوال و كائنات إلخ..

❖ مفهوم الخصائص :

أي متغيرات يتم تعريفها بداخل كلاس و خارج أي دالة تسمى خصائص (Attributes) , و هذا يعني أن أي كائن من هذا الكلاس سيكون عنده هذه الخصائص .
تستطيع التعامل مع هذه الخصائص من الكائن مباشرةً , بينما المتغيرات العادية لا يمكنك التعامل معها من الكائن .

❖ مفهوم الـ Object :

Object : تعني كائن في اللغة العربية. و الكائن عبارة عن نسخة مطابقة لكلاس معين .
بما أن الكائن عبارة عن نسخة من الكلاس, يمكننا القول أنه لا يمكن إنشاء كائن إذا لم يكن هناك كلاس .

إذاً في مفهوم برمجة الكائنات نقوم بإنشاء كلاس معين يسمونه **blue print** أي (النسخة الخام أو النسخة الأصلية) , و بعدها ننشئ نسخة أو أكثر من هذا الكلاس و نفعل بها ما نريد بدون أن نغير محتويات الكلاس الأساسي و هكذا نكون حافظنا على كودات الكلاس الأساسي لأننا نعدل على النسخ و ليس عليه مباشرةً .

• طريقة التعامل مع الكائنات :

- نقوم بإنشاء كائن من الكلاس .
- بعدها نقوم بإدخال قيم لخصائصه , إستدعاء دواله إلخ..

• لاستدعاء أي شيء موجود في الكائن الذي خلقناه :

- نضع إسم الكائن.
- ثم نقطة.
- ثم الشيء الذي نريد الوصول إليه (سواء إسم متغير أو دالة).

• نصائح عليك إتباعها :

- يفضل إنشاء كل كلاس في ملف جافا خاص.
- إبدأ إسم الكلاس دائماً بحرف كبير.
- إبدأ إسم الكائن دائماً بحرف صغير.

❖ الكلمة this :

الكلمة **this** هي كلمة محبوزة في لغة جافا , و هي تستخدم للإشارة إلى الـ Global Variables, و تستخدم أيضاً للإشارة إلى الكائن الحالي , و يمكن استخدامها في أماكن عديدة .

❖ مفهوم الـ Constructor :

Constructor : تكتب كونستركتور بالعربية .

من أهم الأشياء التي عليك التفكير بها بعد إنشاء كلاس جديد, هي تسهيل طريقة خلق كائنات من هذا الكلاس , من هنا أتت فكرة الكونستركتور وهو عبارة عن دالة لها نوع خاص , يتم إستدعائها أثناء إنشاء كائن لتوليد قيم أولية للخصائص الموجودة فيه .
بما أنه لا يمكن إنشاء كائن من كلاس إلا من خلال كونستركتور , سيقوم مترجم جافا بتوليد كونستركتور افتراضي فارغ عنك إذا وجد أن الكلاس الذي قمت بتعريفه لا يحتوي على أي كونستركتور.

❖ المتغيرات التي يتم وضعها في الكلاس تقسم إلى ثلاث فئات أساسية ذكرناها في الجدول التالي :

Local Variables	هي المتغيرات التي يتم تعريفها بداخل أي دالة , أو constructor , أو بداخل block (مثل الحلقات, الجملة switch إلخ..).
Instance Variables	هي المتغيرات التي يتم تعريفها بداخل الكلاس و خارج حدود أي دالة أو constructor , أو block . تسمى أيضاً Global Variables .
Class Variables	هي المتغيرات التي يتم تعريفها ك static بداخل الكلاس و خارج حدود أي دالة أو constructor , أو block .

❖ نقاط مهمة حول الكونستركتور :

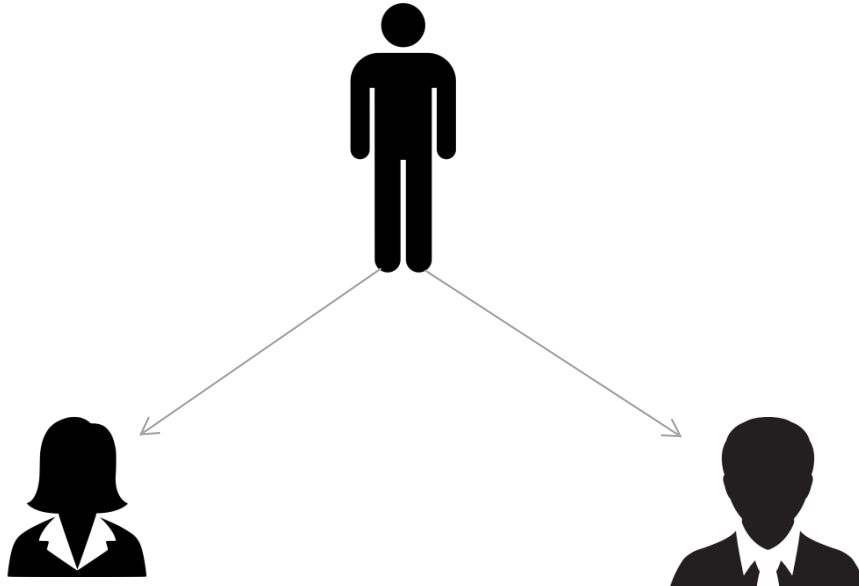
- ✓ كل كلاس يتم إنشاؤه, يحتوي على كونستركتور واحد على الأقل . و حتى إن لم تقم بتعريف أي كونستركتور , سيقوم المترجم بإنشاء واحد افتراضي عنك .
- ✓ في كل مرة يتم إنشاء كائن جديد, يجب استدعاء الكونستركتور .
- ✓ القاعدة الأساسية عند تعريف كونستركتور هي أنه يجب أن يحمل نفس إسم الكلاس و يكون نوعه public.
- ✓ في حال قمت بتعريف كونستركتور , لن يقوم المترجم بإنشاء واحد افتراضي, أي لن يعود هناك كونستركتور افتراضي .
- ✓ يمكنك تعريف أكثر من كونستركتور. و يمكنك دائماً إنشاء كونستركتور فارغ, حتى تستخدمه إن كنت لا تريد إعطاء قيم أولية محددة للخصائص عند إنشاء كائن .

• علاقة ال Object بال Class :

الكائنات تساعد المبرمج كثيراً، فمثلاً إذا كنت تنوي إنشاء برنامج بسيط لحفظ معلومات أشخاص، هل ستنشئ كلاس لكل شخص؟!

طبعاً لا ، بل تنشئ كلاس واحد فقط يمثل شخص ، و تضع فيه الأشياء الأساسية التي تريدها أن تكون موجودة عند كل شخص. ثم تنشئ منه كائنات قدر ما شئت، و عندها يصبح كل كائن من هذا الكلاس عبارة عن شخص له معلوماته الخاصة .

أنا كلاس اسمي Person عبارة عن إنسان وأمتلك الخصائص التالية			
العمر	الوظيفة	الجنس	الإسم



نحن كائنات من الكلاس Person	
الإسم : محمد	الإسم : روز
الجنس : ذكر	الجنس : أنثى
الوظيفة : مهندس	الوظيفة : معلمة
العمر : ٢٦	العمر : ٢٤

الوحدة الحادي عشر

الكائنات والكلاسات Classes and Objects

عملي



❖ لتعريف كلاس جديد يكفي فقط كتابة الكلمة class, ثم وضع إسم له, ثم فتح أقواس تحدد بدايته و نهايته. مثال :

```
class ClassName {  
  
}
```

❖ بما أن الكائن عبارة عن نسخة من الكلاس. لتعريف كائن من كلاس معين يجب وضع إسم الكلاس ثم وضع إسم للكائن .

```
Person ahmad = new Person();
```

- هنا قمنا بتعريف كائن من الكلاس Person إسمه ahmad .
- إذاً الكائن ahmad سيكون عنده نسخة خاصة فيه من خصائص الكلاس Person .

ملاحظة : الكود new Person() هو الذي يقوم فعلياً بتوليد كائن من الكلاس. و هو يعطي قيم أولية للخصائص الموجودة فيه .



بما أنه لا يمكن إنشاء كائن من كلاس إلا من خلال كونسرتكتور، سيقوم مترجم جافا بتوليد كونسرتكتور افتراضي فارغ عندك إذا وجد أن الكلاس الذي قمت بتعريفه لا يحتوي على أي كونسرتكتور .

إذا قمنا بتعريف كلاس اسمه Person و لم نقم بتعريف كونسرتكتور له كما في الكلاس التالي

```
class Person {  
  
}
```

سيقوم المترجم بإنشاء كونسرتكتور فارغ بشكل تلقائي عننا كالتالي

```
class Person {  
  
    public Person() {  
  
    }  
  
}
```

الآن سنقوم بإنشاء الكلاس Person و إنشاء كائنات منه في الكلاس الذي يحتوي على الدالة main().

ملاحظة : يجب إنشاء الكلاس Person و الكلاس Main في نفس الـ package حتى يعمل الكود بشكل صحيح .



Person.java

```
public class Person {  
  
    // هنا قمنا بتعريف ٤ خصائص  
    String name;  
    String sex;  
    String job;  
    int age;  
  
    // هنا قمنا بتعريف دالة تطبع محتوى كل خاصية عندما يتم  
    // استدعاءها  
    void printInfo() {  
        System.out.println("Name: " +name);  
        System.out.println("Sex: " +sex);  
        System.out.println("Job: " +job);  
        System.out.println("Age: " +age);  
        System.out.println();  
    }  
}
```

Main.java

```
public class Main {  
  
    public static void main(String[] args) {  
  
        // هنا قمنا بإنشاء كائنات من الكلاس Person  
        Person p1 = new Person(); // الكائن p1 سيمثل محمد  
        Person p2 = new Person(); // الكائن p2 سيمثل روز  
        Person p3 = new Person(); // الكائن p3 سيمثل أحمد  
        Person p4 = new Person(); // الكائن p4 سيمثل ربيع  
  
        // هنا قمنا بتحديد خصائص الكائن p1  
        p1.name = "Mhamad";  
        p1.sex = "Male";  
        p1.job = "Programmer";  
        p1.age = 21;  
  
        // هنا قمنا بتحديد خصائص الكائن p2  
        p2.name = "Rose";  
        p2.sex = "Female";  
        p2.job = "Secretary";  
        p2.age = 22;  
    }  
}
```



```
// هنا قمنا بتحديد خصائص الكائن p3
p3.name = "Ahmad";
p3.sex = "Male";
p3.job = "Doctor";
p3.age = 34;

// هنا قمنا بتحديد خصائص الكائن p4
p4.name = "Rabih";
p4.sex = "Male";
p4.job = "Engineer";
p4.age = 27;

// هنا قمنا بعرض خصائص كل كائن
p1.printInfo();
p2.printInfo();
p3.printInfo();
p4.printInfo();

}

}
```

سنحصل على النتيجة التالية عند التشغيل

Name: Mhamad
Sex: Male
Job: Programmer
Age: 21

Name: Rose
Sex: Female
Job: Secretary
Age: 22

Name: Ahmad
Sex: Male
Job: Doctor
Age: 34

Name: Rabih
Sex: Male
Job: Engineer
Age: 27

تمارين عامة + مراجعة شاملة

تمارين



اسم المتدرب :

الرقم التدريبي :

❖ إجابات السؤال الأول : (T) اذا كانت الإجابة صحيحة (F) إذا كانت الإجابة خاطئة :-

١٠	٩	٨	٧	٦	٥	٤	٣	٢	١
٢٠	١٩	١٨	١٧	١٦	١٥	١٤	١٣	١٢	١١
٣٠	٢٩	٢٨	٢٧	٢٦	٢٥	٢٤	٢٣	٢٢	٢١

❖ إجابات السؤال الثاني : (اختيارات A,B,C,D) :-

١٠	٩	٨	٧	٦	٥	٤	٣	٢	١
٢٠	١٩	١٨	١٧	١٦	١٥	١٤	١٣	١٢	١١
٣٠	٢٩	٢٨	٢٧	٢٦	٢٥	٢٤	٢٣	٢٢	٢١

السؤال الأول : ضع علامة (✓) امام العبارة الصحيحة وعلامة (x) امام العبارة الخاطئة :

١. البرمجيات (**Software**) هي التي تسهل للمستخدم استخدام المكونات المادية
٢. برامج التشغيل هي برامج تساعد في إنشاء كثير من التطبيقات مثل إنشاء قاعدة البيانات والرسم بإستخدام الحاسب .
٣. برنامج الحاسب هو عبارة عن مجموعة من التعليمات تعطى للحاسب للقيام بعمل ما
٤. لغات البرمجة يمكن تقسيمه إلى أربعة أنواع رئيسية
٥. لغة الجافا تعتبر من لغات التجميع
٦. من برامج التشغيل الاكسل والاكسس والأوراكل
٧. لغة الآلة تتكون من مجموعة أرقام بين ٠ , ١
٨. خرائط التدفق **Flow Charts** هو تمثيل رسومي للخوارزمية
٩. من أنواع خرائط التدفق خرائط سير النظم وخرائط سير البرامج
١٠. الخرائط ذات الفروع من خرائط سير البرامج
١١. خرائط الدوران الواحد من خرائط سير النظم
١٢. الخوارزميات هي مجموعة من الخطوات والتعليمات المرتبة لتنفيذ عملية حسابية أو منطقية أو غيرها بشكل متتابع متسلسل ومنظم
١٣. من برامج التطبيقات الأوراكل والفوتوشوب والأوتوكاد
١٤. تعتبر لغة الجافا بسيطة وسهلة ولها شعبية هائلة

١٥. لغة الجافا تعتبر من اللغات مفتوحة المصدر
١٦. لغة الجافا لا تميز بين الحروف الكبيرة والصغيرة
١٧. الفاصلة المنقوطة (**Semicolon**) تستعمل عند نهاية كل أمر في برامج الجافا
١٨. من قواعد تسمية الأسماء في الجافا أن يبدأ برقم
١٩. نستخدم التعليقات في الجافا لنضع ملاحظات حول الكود الذي كتبناه فقط
٢٠. كلمة **Long** من الكلمات المحجوزة في لغة الجافا
٢١. المتغيرات هو مكان يتم حجزه في الذاكرة لتخزين بيانات أثناء تشغيل البرنامج
٢٢. **Double** هذا النوع يمثل عدد بفاصلة عشرية يتألف من ٦٤ bit
٢٣. لغة الجافا تحتوي على طريقة واحدة فقط لإخراج أو طباعة البيانات
٢٤. العامل **&&** يستخدم لتنفيذ كود معين إذا لم يتحقق الشرط الأول و الشرط الثاني
٢٥. العامل **||** يستخدم لتنفيذ كود معين إذا تحقق على الأقل واحد من الشروط الموضوعة
٢٦. العامل **!** يستخدم لتنفيذ كود معين إذا لم يتحقق أي شرط تم وضعه
٢٧. جملة **if** أبسط الجمل الشرطية فهي تحتوي على شرط , عند تحققه ينفذ أمر معين
٢٨. الجملة الشرطية **IF** هي أحد أهم الأدوات في البرمجة بشكل عام
٢٩. جمل الدوران في الجافا تستخدم لتكرار كتلة من التعليمات البرمجية حتى يتم استيفاء شرط معين
٣٠. نستخدم الحلقة **for** إذا كنا نريد تنفيذ الكود عدة مرات محددة



السؤال الثاني : ضع دائرة حول الاجابة الصحيحة :

١. من برامج التشغيل :

- A. الأوراكل Oracle
- B. DOS
- C. الأوتوكاد Autocad
- D. الفوتوشوب Photoshop

٢. من أمثلة لغات البرمجة :

- A. الاكسيل Excel
- B. Linux
- C. VMS
- D. الجافا Java

٣. من الكلمات المحبوزة في لغة جافا :

- A. True
- B. Eng
- C. Ali
- D. Omg

٤. يتم تعريف متغير عدد عشري في لغة الجافا عن طريق الامر :

- A. Double
- B. Int
- C. Boolean
- D. String

٥. يعتبر من لغات البرمجة ذات المستوى العالي :

- A. الباسكال Pascal
- B. الاكسيل Excel
- C. الأوتوكاد Autocad
- D. الفوتوشوب Photoshop



٦. هي برامج تساعد في إنشاء كثير من التطبيقات مثل إنشاء قاعدة بيانات والرسم باستخدام الحاسب :

- A. برامج التطبيقات
- B. برامج التشغيل
- C. لغات البرمجة
- D. لغة الآلة

٧. يتم تعريف متغير رقمي في لغة الجافا عن طريق الامر :

- A. Double
- B. int
- C. Boolean
- D. String

٨. من برامج التطبيقات :

- A. الأكسس Access
- B. ويندوز Windows
- C. Dos
- D. البيسك Basic

٩. تتكون من مجموعة أرقام من بين ٠ , ١ :

- A. لغة الآلة
- B. لغة التجميع
- C. لغات ذات المستوى العالي
- D. لغة الجافا

١٠. المقارنة (a & b) يدل على :

- A. OR
- B. NOT
- C. AND
- D. لا شيء مما ذكر



١١. هي عبارة عن برامج تقوم بدور الوسيط بين المستخدم والمكونات المادية وهي تمكن المستخدم من استخدام المكونات المادية للحاسب بكفاءة وراحة , هذا تعريف :

- A. برنامج الحاسب
- B. برامج التطبيقات
- C. لغات البرمجة
- D. برامج التشغيل

١٢. هي مجموعة من الخطوات والتعليمات المرتبة لتنفيذ عملية حسابية أو منطقية أو غيرها بشكل متتابع متسلسل ومنظم , هذا تعريف :

- A. الخوارزميات
- B. خرائط التدفق
- C. تقسيم المشكلة
- D. البرمجيات

١٣. يعتبر من أنواع الجمل الشرطية :

- A. Switch
- B. While
- C. Do While
- D. for

١٤. يعتبر من أنواع جمل الدوران :

- A. if
- B. Switch
- C. for
- D. Else

١٥. العامل الذي يستخدم في وضع الشروط المنطقية :

- A. go
- B. And
- C. true
- D. else

١٦. المقارنة ($a < b$) يدل على :

- A. أكبر من
- B. أكبر من أو يساوي
- C. أصغر من
- D. أصغر من أو يساوي

١٧. رمز العامل OR :

- A. &&
- B. \$\$
- C. !
- D. ||

١٨. رمز العامل And :

- A. &&
- B. \$\$
- C. !
- D. ||

١٩. رمز العامل NOT :

- A. &&
- B. \$\$
- C. !
- D. ||

٢٠. جملة (if) يتم استخدامها اذا كان لدينا :

- A. احتمالين
- B. أربع احتمالات
- C. احتمال واحد
- D. ثلاث احتمالات



٢١. جملة (else) يتم استخدامها اذا كان لدينا :

- A. احتمالين
- B. أربع احتمالات
- C. احتمال واحد
- D. ثلاث احتمالات

٢٢. تستخدم لتكرار كتلة من التعليمات البرمجية حتى يتم استيفاء شرط معين :

- A. الخوارزميات
- B. جمل الدوران
- C. جمل الشرطية
- D. لا شيء مما ذكر

٢٣. نستخدم الجملة التالية لتجاوز تنفيذ كود معين في الحلقة :

- A. Continue
- B. if
- C. Switch
- D. لا شيء مما ذكر

٢٤. نستخدم الحلقة التالية إذا كنا نريد تنفيذ الكود عدة مرات محددة :

- A. For
- B. if
- C. Switch
- D. لا شيء مما ذكر

٢٥. جملة (Switch) تتكون من :

- A. جزء واحد
- B. جزئين
- C. ثلاث أجزاء
- D. لا شيء مما ذكر



٢٦. عن طريق هذا الأمر نقوم بمعالجة كل حاله لوجودها بشكل منفصل :

break .A

Case .B

default .C

cmd .D

٢٧. تعتبر من جمل التحكم في الحلقات :

Continue .A

if .B

Switch .C

.D لا شيء مما ذكر

٢٨. المقارنة (a || b) يدل على :

OR .A

NOT .B

AND .C

.D لا شيء مما ذكر

٢٩. المقارنة (! a) يدل على :

OR .A

NOT .B

AND .C

.D لا شيء مما ذكر

٣٠. تعتبر من الكلمات المحبوزة في لغة الجافا :

are .A

new .B

To .C

.D لا شيء مما ذكر



السؤال الثالث : قم بكتابة الخوارزميات وخرائط التدفق لبرنامج يقوم بجمع عددين ومن ثم طباعة الناتج النهائي ؟

طريقة خرائط التدفق	طريقة الخوارزميات

السؤال الرابع : قم بكتابة الخوارزميات وخرائط التدفق لبرنامج يقوم بقراءة درجة الطالب في الاختبار فإذا كانت درجته أكبر من أو يساوي ٦٠ يخرج كلمة (ناجح) و إذا كانت أقل من ذلك يخرج البرنامج كلمة (راسب) ؟

طريقة خرائط التدفق	طريقة الخوارزميات

السؤال الخامس : صحح الأكواد التالية ؟

الكود	تصحيح الكود
<pre>String Num1 ; int num2 ; int sum ; Num1 = 30 Num2 = 20 ; Num = Num1 + Num2 ; System.out.println("sum =" + Num) ;</pre>	

الكود	تصحيح الكود
<pre>String a ; a= 3100 if (a <= 3000) ; System.out.println("تستحق الدعم") ; else System.out.println(لا تستحق الدعم) ;</pre>	



السؤال السادس : اكتب مخرجات البرامج التالية ؟

المخرجات	البرنامج
	<pre>int a ; a= 18 ; if (a > 18) System.out.println("عاقل") ; else System.out.println("غير عاقل") ;</pre>

المخرجات	البرنامج
	<pre>int a ; a = -60 ; if (a >= 60 && a <= 100) System.out.println("مجتاز") ; else if (a < 60 && a >= 0) System.out.println("غير مجتاز") ; else System.out.println("الدرجة غير معروفة") ;</pre>

المخرجات	البرنامج
	<pre>for(int i=1; i<=10; i++) { if(i%2 == 0) continue; System.out.println(i); }</pre>

السؤال السابع : اكتب برنامج يقوم بجمع عددين , العدد الأول = 30 والعدد الثاني = 20

ثم يقوم بطباعة مجموعهما :

الكود	
	المخرجات



السؤال الثامن : اكتب برنامج يقوم المستخدم بإدخال الدرجة , إذا كانت الدرجة 90 وأعلى يطبع "ممتاز" إذا كانت الدرجة 80 وأعلى يطبع " جيد جدا " , إذا كانت الدرجة 70 وأعلى يطبع " جيد " إذا كانت الدرجة 60 وأعلى يطبع " مقبول " , غير ذلك يطبع " راسب "

الكود	
	المخرجات

مصطلحات و اختصارات الكيبورد



مصطلحات



مصطلحات شائعة

تغيير	change	خاص	private
مساعدة	help	عام	public
تشغيل	run	مبرمج	programmer
تحميل	Download	برنامج	program
تحديث	refresh	برمجيات	Software
بحث	search	المعدات	hardware
مفتوح المصدر	Open source	أنظمة تشغيل	operating systems
إنهاء	exit	برامج تطبيقات	Software programs
إلغاء الأمر	cancel	لغات برمجة	Programming Languages
مجموع	sum	لغة الآلة	Machine language
ملف	file	مترجم	compiler
طباعة	print	مفسر	interpreter
نظام	system	مشكلة	problem
تعليق	comment	حل	solving
تكرار أو دوران	looping	الخوارزمية	algorithm



نهاية	stop	منطوق	flowchart
بداية	start	خطأ	error
بيانات	data	صحيح	true
إخراج	output	إدخال	input
دوال	Methods	تقرير	report
شرط	condition	كائن	object
بيان	statement	دالة	Function
تهيئة	Initialization	الطول	length
إنقاص	decrement	مثال	example
زيادة	increment	مصفوفة	Array
استراحة	break	إفتراضي	default
معلومات	informations	تعبير	expression
رسالة	Message	تحذير	Warning
خيارات	Options	لا شيء	null
المتغيرات	Variables	أنواع العمليات	Operations Types
سلسلة	Concatenation	معالجة	Process
مشروع	Project	فاصلة منقوطة	Semicolon

اختصارات الكمبيوتر

حفظ	ctrl+s
نسخ	ctrl+c
لصق	ctrl+v
قص	Ctrl + X
تراجع	Ctrl + Z
يحول الكتابة من عربي إلى إنجليزي	alt+shift
}	Shift + V
{	Shift + C
]	Shift + D
[Shift + F
تحديد الكل	Ctrl + A
ملف جديد	Ctrl + N
طباعة	Ctrl + P
فتح ملف	Ctrl + O
تمديد الحرف	Shift + j
مسافة مفردة	Ctrl + 1
مسافة سطر ونصف	Ctrl + 5
مسافة مزدوجة	Ctrl + 2
الانتقال لنهاية الملف	Ctrl + END
علامة تنصيص "	Ctrl + ط
تكبير النص	Ctrl + د
تصغير النص	Ctrl + ج
أول المستند	Ctrl + Home

المراجع



Introduction to Java programming
Mohammed Harmoush educational site
Curriculum General Administration
Y. Daniel Liang
The basics of the Java language
Introduction to the Java Language
Algorithms and logic
Introduction to the world of programming



م. عبدالمجيد بن عيظه العتيبي

هندسة حاسب آلي

هذا الكتاب إهداء إلى من ...

“ يطمح أن يكون علامة فارقة ”

للتواصل

mjeed_only 

| info@mjeed.net 

| www.mjeed.net 

تَمَّ بِخَفْدِ اللَّهِ