محاضرات العملي لمقرر

البرمجة الشيئية

ITGS211

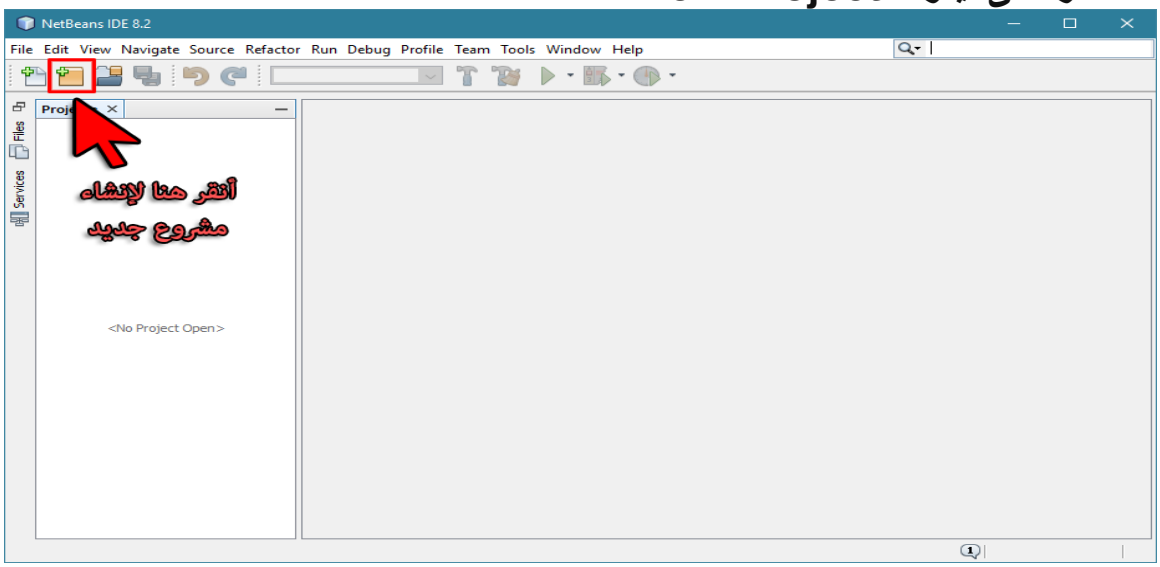
الدرس العملي رقم (1)

إعداد الأستاذة :ملاك ددش

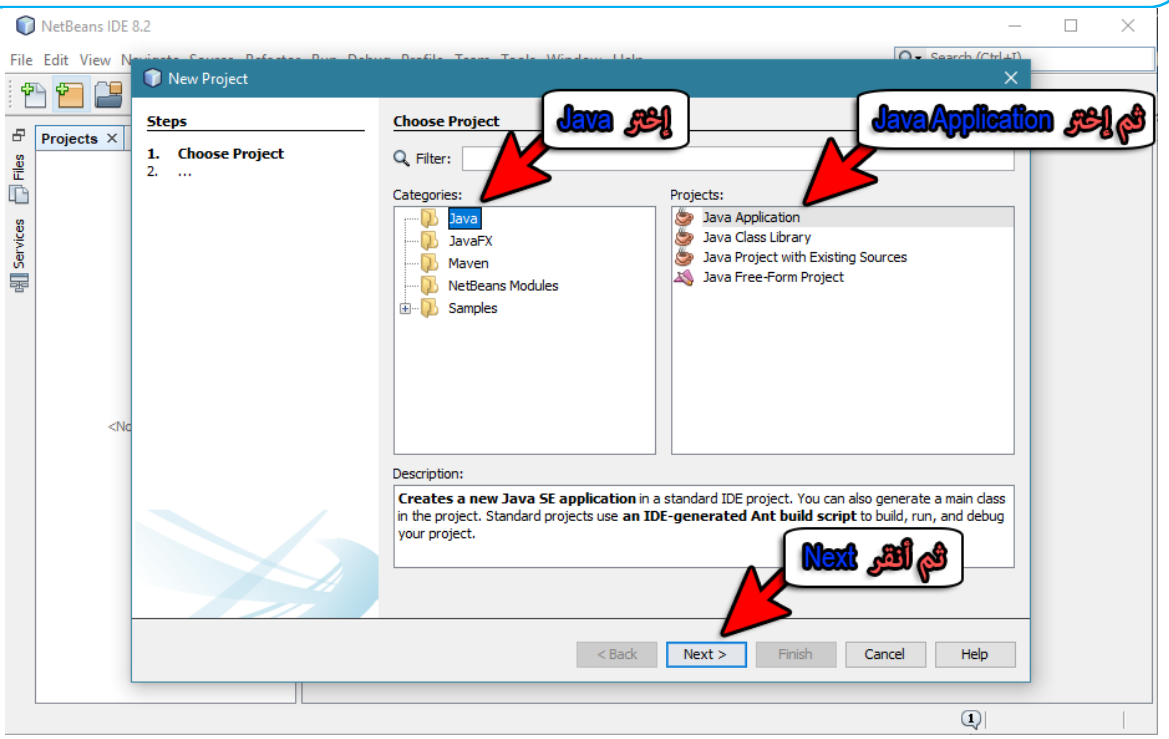
- **في الجزء العملي سنعتمد برنامج NetBeans لأنه من أفضل و أقوى البرامج المجانية المتاحة لكتابة كود بلغة جافا.:NetBeans**
- **تعليمات تثبيت Java and NetBeans :Java and NetBeans installation guide**

خطوات إنشاء مشروع جديد و تشغيله في برنامج NetBeans:

1أنقر على أيقونة New Project



2 ثم أنقر على مجلد Java, ثم مجلد Java Application, ثم أنقر على الزر Next كما في الصورة التالية



3 الآن قم بوضع إسم للمشروع, ثم ضع إسم للكلاس الذي سيتم إنشاؤه بداخل المشروع, ثم إضغط Finish كما في الصورة التالية.

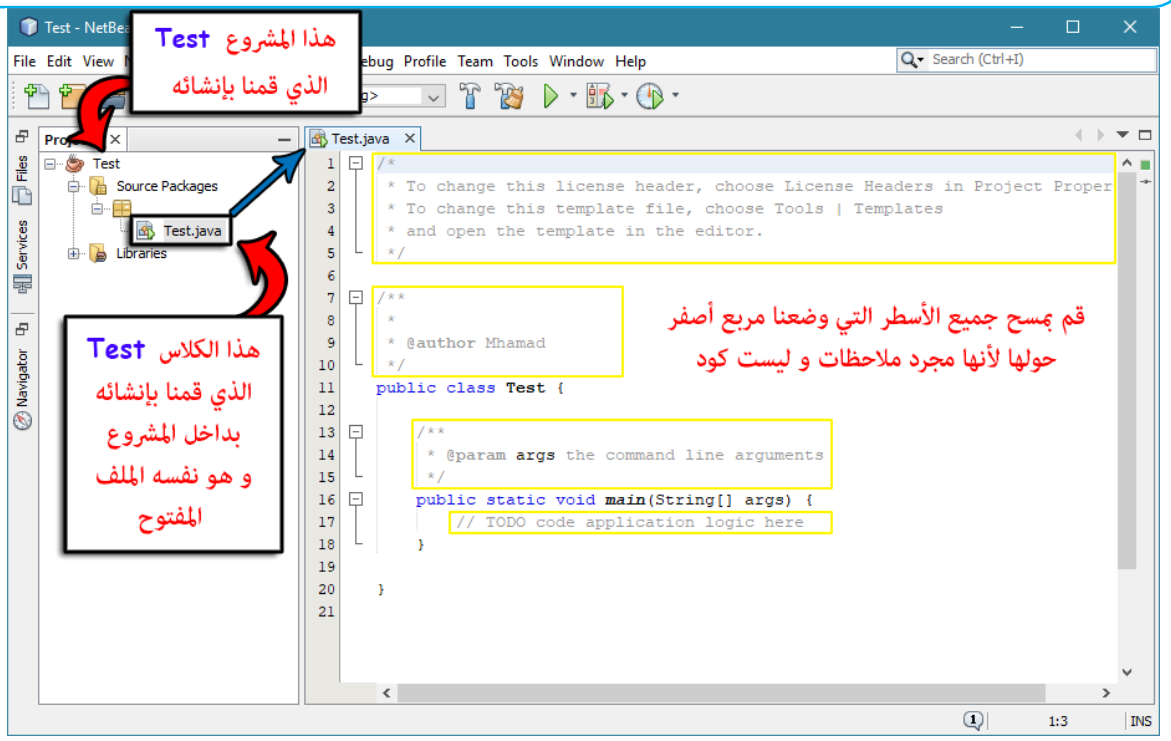
ملاحظة :هنا قمنا بتسمية المشروع و الكلاس الذي بداخله Test مع الإشارة إلى أنه يمكنك وضع أي إسم تريد.

إنتبه :لا يمكنك إنشاء مشروعين عندهم نفس الإسم و إن فعلت ذلك سيظهر لك أن هناك مشكلة.

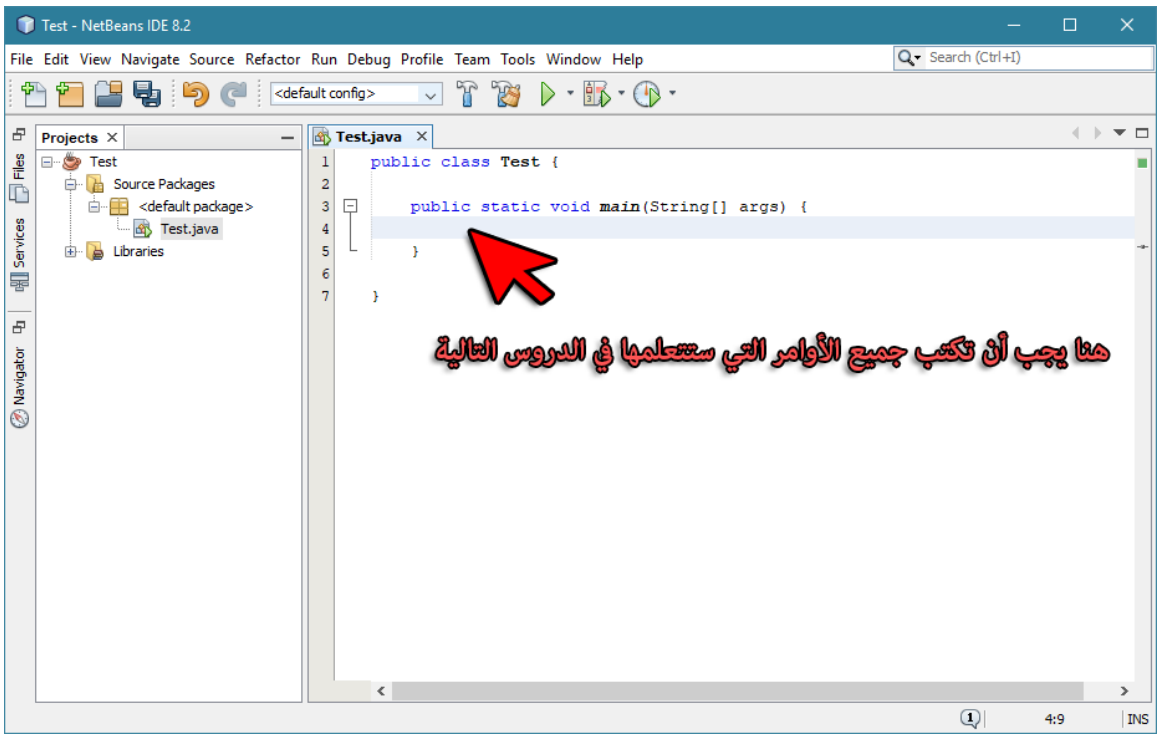
4 قم بمسح جميع التعليقات الموجودة في ملف الجافا بتأني و إنتبه جيداً لكي لا تمسح كودات الجافا.

ملاحظة :كل شيء موضوع بين /* *أو يوجد قبله // هو مجرد تعليق.

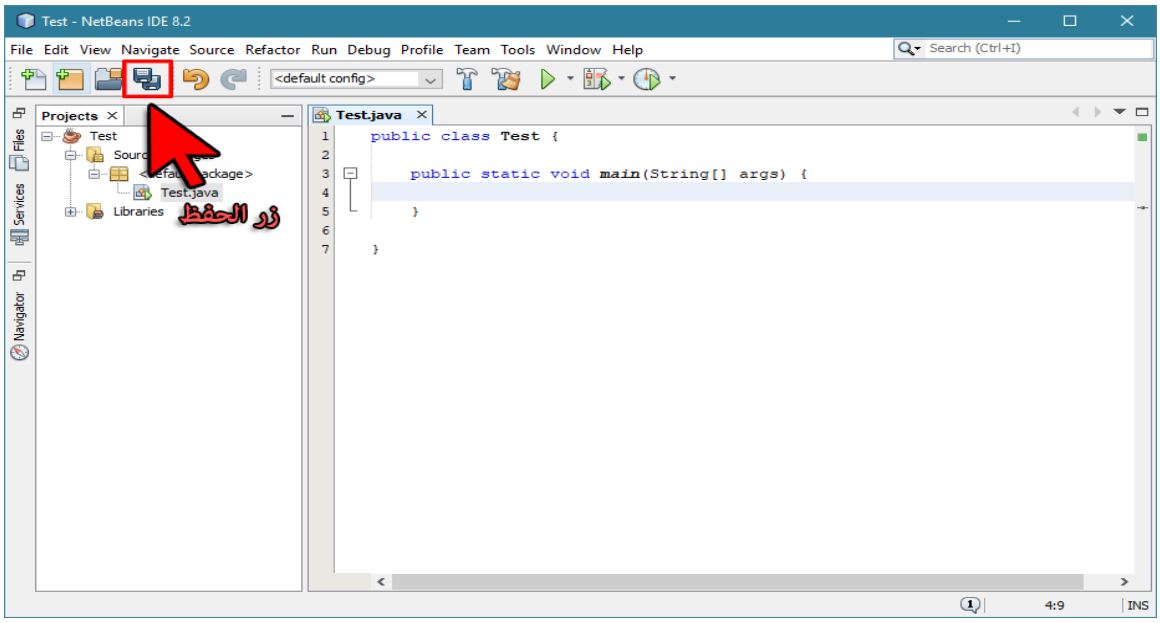
التعليق هو ملاحظة يكتبها المبرمج فوق الكود و هذه الملاحظة لا تؤثر إطلاقاً على الكود, لذلك قم بمسح جميع التعليقات عند إنشاء أي مشروع جديد لأنك لست بحاجة إليهم



5 بعد مسح جميع التعليقات سيكون شكل الكلاس هكذا



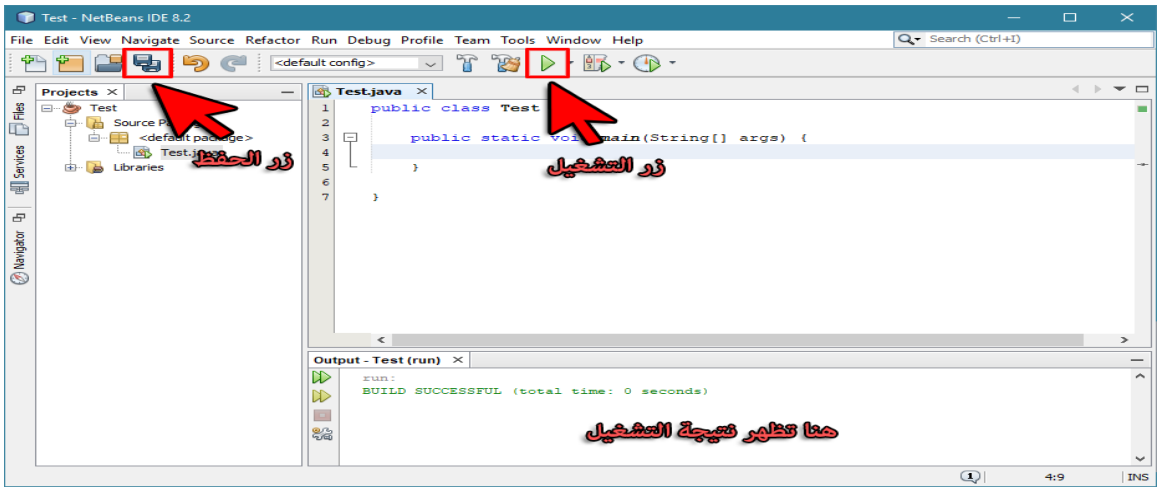
6 في الأخير إضغط على أيقونة الحفظ لتحفظ جميع التعديلات التي قمت بإجرائها.



خطوات تشغيل المشروع:

1 في الدروس اللاحقة عندما تنتهي من كتابة أي برنامج, إضغط على أيقونة الحفظ ثم أيقونة التشغيل لكي يقوم بتشغيل البرنامج الذي قمت بكتابته.

2 عندما تقوم بتشغيل البرنامج ستظهر نافذة جديدة في أسفل الشاشة و تظهر فيها نتيجة تشغيل الكود الذي قمت بكتابته, و كم ترى أنه لا يظهر لك أي شيئ, لأنك لم تكتب أي كود بعد.



الشكل العام لأي برنامج مكتوب بلغة جافا

```
public class هنا يوضع إسم الكلاس {

    public static void main(String[] args) {

        هنا يجب أن تضع الأوامر التي ستنتفذ عند تشغيل البرنامج

    }

}
```

الكود الذي يتنفذ مباشرةً عند تشغيل البرنامج هو الكود الذي نكتبه بداخل حدود الدالة main().

في المثال التالي قمنا بإنشاء برنامج إسمه Main, مهمته فقط طباعة الجملة "Hello World!" عند التشغيل.

مثال

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- سنحصل على النتيجة التالية عند التشغيل.

```
Hello World!
```

المصطلحات الأساسية في لغة جافا هي التالية:

- **الكلاس (Class)**
- **الكائن (Object)**
- **الخصائص (Instance variables)**
- **الدوال (Methods)مبادئ كتابة الكود في جافا**

مبادئ كتابة الكود في جافا:

1 إحترام أشكال الأحرف:

لغة جافا تطبق مفهوم Case Sensitivity, و هذا يعني أنها تميّز بين الأحرف الكبيرة و الأحرف الصغيرة.

مثال: note و Note ليسوا شيئاً واحداً.

2 إسم الكلاس:

دائماً إبدأ إسم الكلاس بحرف كبير.
مثال: class First

إذا كان إسم الكلاس يتألف من أكثر من كلمة, إجعل أول حرف من كل كلمة كبيراً.
مثال: class FirstJavaClass

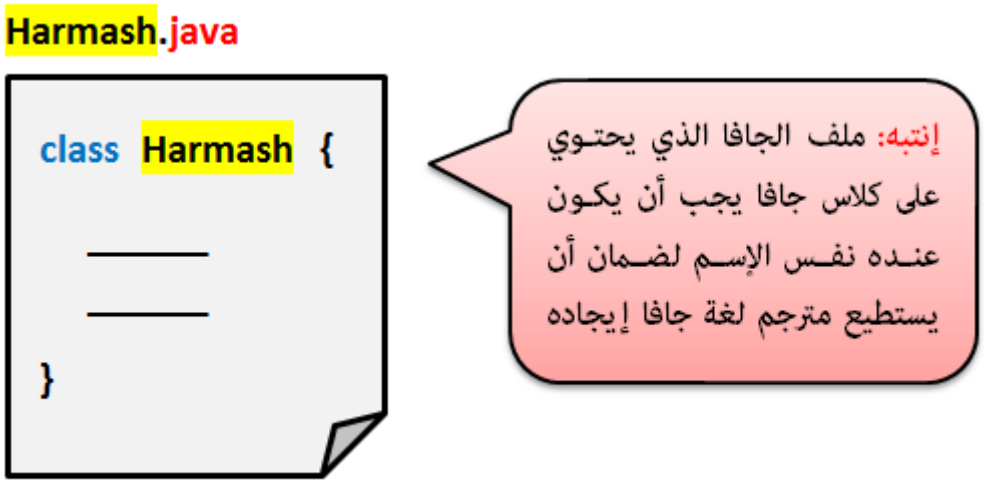
3 اسم الدالة:

دائماً إبدأ إسم الدالة بحرف صغير.
مثال: public void display()

إذا كان إسم الدالة يتألف من أكثر من كلمة, إجعل أول كلمة تبدأ بحرف صغير و جميع الكلمات التي تليها تبدأ بحرف كبير.
مثال: public void displayUserInfo()

4 إسم الملف الذي يحتوي على الكلاس:

إسم كلاس الجافا يجب أن يكون مطابقاً لإسم الملف الموجود بداخله.
مثال: إذا كان إسم الكلاس class Harmash يجب أن يكون موجوداً في ملف جافا إسمه Harmash.java كما في هذه الصورة:



نقطة البداية:

نقطة البداية (Entry Point)في جافا هي الدالة ()main و لذلك عند إنشاء أي مشروع جديد تجد هذه الدالة جاهزة فيه كالتالي:

مثال

```
public class Example {

    // main() الدالة
    public static void main(String[] args) {

        // الكود الذي يوضع هنا هو الذي سيتنفذ عند تشغيل البرنامج

    }

}
```

إذاً, بالمنطق يجب أن يحتوي أي مشروع على دالة ()main واحدة لأنها نقطة البداية.
و كخلاصة للموضوع, نقطة البداية تعني أول مكان سيتم تنفيذ الأوامر الموجودة فيه عند تشغيل البرنامج.

التعليقات:

تعليق على سطر واحد.

مثال

```
// هذا تعليق يتألف من سطر واحد
```

تعليق على عدة أسطر.

مثال

```
/* هذا تعليق
 * يتألف من
 * عدة أسطر
 */
```

الكلمات المحجوزة في جافا:

جميع الكلمات التالية محجوزة للغة جافا, أي لا يمكن استخدامها identifiers

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