بسم الله الرحمن الرحيم

لغة البرمجة جافا

Java Programming Language

الدرس الثامن :

الكلمة المفتاحية

final

الكلمة المفتاحية final :

كلمة final تأخذ عدة معاني حسب السياق و حسب الاستخدام و لكن بشكل عام تعني " الشي لا يتغير "

أولا : كلمة final مع البيانات :

كل لغات البرمجة لها أساليب لتحديد أن متحول ما هو ثابت . في جافا تستخدم كلمةfinal للدلالة على أن متحول ما هو ثابت . يجب إعطاء القيمة في المكان الذي تعرف فيه

عندما تستخدم final مع الأغراض فإنها تأخذ معنى مختلف عند استخدامها مع المتحولات .

مع المتحولات كلمة final تجعل القيمة ثابتة

مع الأغراض كلمة final تجعل العنوان ثابت و عندما يهيأ هذا العنوان بغرض فإنه لا يمكن تغييره و لكن الغرض نفسه يمكن تغييره .

لغة جافا لا تملك طريقة لجعل غرض ما ثابت

هناك متحولات و أغراض مع final ..

Compile time final : تأخذ قيمها أثناء الترجمة

Run time final : تأخذ قيمتها عند التنفيذ عن طريق تعبير

مثال :

```
class Value {
      int i = 1;
}

public class FinalData {
      final int i1 = 9; // compile time ...
      static final int VAL_TWO = 99;
      public static final int VAL_THREE = 39;

      final int i4 = (int)(Math.random()*20); // run time
      static final int i5 = (int)(Math.random()*20);
      Value v1 = new Value();
      final Value v2 = new Value();
      static final Value v3 = new Value();

      final int[] a = { 1, 2, 3, 4, 5, 6 };

      public void print(String id) {
            System.out.println(
            id + ": " + "i4 = " + i4 +
            ", i5 = " + i5);
```

```
	}

	public static void main(String[] args) {
		FinalData fd1 = new FinalData();
		//! fd1.i1++; // Error: can't change value
		fd1.v2.i++; // Object isn't constant!
		fd1.v1 = new Value(); // OK -- not final
		for(int i = 0; i < fd1.a.length; i++)
			fd1.a[i]++; // Object isn't constant!
		//! fd1.v2 = new Value(); // Error: Can't
		//! fd1.v3 = new Value(); // change reference
		//! fd1.a = new int[3];
		fd1.print("fd1");
		System.out.println("Creating new FinalData");
		FinalData fd2 = new FinalData();
		fd1.print("fd1");
		fd2.print("fd2");
	}
}
```

Blank final :

تسمح لغة جافا بالتصريح عن متحولات أو أغراض ثابتة بدون إعطاء قيم ابتدائية لهذه المتحولات أو الأغراض .

ولكن هذه المتحولات او الأغراض يجب أن تأخذ قيم قبل الاستخدام

مثال :

```
class P {
}

class BlankFinal {
	final int i = 0; // Initialized final
	final int j; // Blank final
	final P p; // Blank final reference

	// Blank finals MUST be initialized
	// in the constructor:
	BlankFinal() {
		j = 1; // Initialize blank final
		p = new P();
	}

	BlankFinal(int x) {
		j = x; // Initialize blank final
		p = new P();
	}
```

```
      public static void main(String[] args) {
            BlankFinal bf = new BlankFinal();
      }
}
```

المترجم يجبرنا أن نعطي القيم الابتدائية للحقول الثابتة إما عن طريق تعبير في مكان التعريف أو في الباني و بذلك يضمن أن الحقل الثابت قد هيئ بشكل صحيح .

البارمترات الثابتة (final argument):

عند التصريح عن بارمتر أنه ثابت فذلك يعني أنه لا يمكن تغيير قيمته داخل جسم التابع

مثال :

```
class G {
      public void spin() {}
}

public class FinalArguments {
      void with(final G g) {
            //! g = new G(); // Illegal -- g is final
      }

      void without(G g) {
            g = new G(); // OK -- g not final
            g.spin();
      }

      // void f(final int i) { i++; } // Can't change
      // You can only read from a final primitive:
      int g(final int i) { return i + 1; }

      public static void main(String[] args) {
            FinalArguments bf = new FinalArguments();
            bf.without(null);
            bf.with(null);
      }
}
```

تتم قراءة الكود و مناقشته ..

الطرق الثابتة (final method):

السبب الذي يجعلنا نجعل طريقة ما ثابتة هو أن نمنع عمليات إعادة تعريفها في الصفوف الأبناء .

الطريقة التي تكون private بشكل ضمني تكون final . حيث أننا لا نستطيع الوصول للطريقة التي تكون private و بالتالي لا يمكن إعادة تعريفها .

يمكن إضافة كلمة final لطريقة private و لكن ذلك لا يضيف أي معنى .

مثال :

```
class WithFinals {
      private final void f() {
            System.out.println("WithFinals.f()");
      }
      // Also automatically "final":
      private void g() {
            System.out.println("WithFinals.g()");
      }
}

class OverridingPrivate extends WithFinals {
      private final void f() {
            System.out.println("OverridingPrivate.f()");
      }

      private void g() {
            System.out.println("OverridingPrivate.g()");
      }
}

class OverridingPrivate2 extends OverridingPrivate {
      public final void f() {
            System.out.println("OverridingPrivate2.f()");
      }

      public void g() {
            System.out.println("OverridingPrivate2.g()");
      }
}

public class FinalOverridingIllusion {
      public static void main(String[] args) {
            OverridingPrivate2 op2 =
            new OverridingPrivate2();
            op2.f();
            op2.g();
            // You can upcast:
            OverridingPrivate op = op2;
            // But you can't call the methods:
            //! op.f();
            //! op.g();
            // Same here:
```

```
            WithFinals wf = op2;
            //! wf.f();
            //! wf.g();
      }
}
```

تتم مناقشة الكود و شرح معنى واجهة بشكل مختصر ..

الصفوف الثابتة (final class) :

عندما نقول عن صف أنه ثابت فذلك يعني أننا لا نستطيع أن نرث منه ..

مثال :

```
class SmallBrain {}

final class Dinosaur {
      int i = 7;
      int j = 1;
      SmallBrain x = new SmallBrain();

      void f() {}
}

//! class Further extends Dinosaur {}
// error: Cannot extend final class 'Dinosaur'
public class Jurassic {
      public static void main(String[] args) {
            Dinosaur n = new Dinosaur();
            n.f();
            n.i = 40;
            n.j++;
      }
}
```

ساهم بنشر الكتاب

لا تنسوني من صالح دعائكم

تم بحمد الله