

Tema: I Presentació.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant

Curs 2014-2015 , Copyleft © 2011-2015 .
Reproducció permesa sota els termes de la llicència de documentació
lliure GNU.



1 / 44

Professors

- Garrigós Fernández, Irene - (Coordinadora)
- Corbí Bellot, Antonio-Miguel
- Muñoz Terol, Rafael
- Martínez-Larraz Prats, Carlos

Despatxos, Horaris de tutoria, cita prèvia, etc: www.dlsi.ua.es.



3 / 44

Contingut

- 1 Professors
- 2 Guia Docent
- 3 Continguts
- 4 Avaluació
- 5 Avaluació sense superar avaluació contínua
- 6 Pla d'aprenentatge (I)
- 7 Pla d'aprenentatge (II)
- 8 Llenguatges de pràctiques
- 9 Introducció a Vala
- 10 Característiques de Vala
- 11 Paraules reservades
- 12 Operadors
- 13 Hola Món en Vala
- 14 Compilació
- 15 Cadenas
- 16 Entrada / Sortida
- 17 Arrays
- 18 Classes



2 / 44

Guia Docent

- **Campus Virtual** → **Recursos d'aprenentatge** → **Guia docent**
- Horaris, objectius i competències, continguts, pla de aprenentatge, avaluació, bibliografia i enllaços
- L'assignatura proporciona 1.20 crèdits teòrics i 1.20 crèdits pràctics.



4 / 44

El temari de l'assignatura és el següent

- **T1** - Presentació, Llenguatges de programació
- **T2** - Control de versions
- **T3** - Programació dirigida per esdeveniments i execució diferida de codi
- **T4** - Interfícies gràfiques d'usuari
- **T5** - Accés a BBDD des d'aplicacions d'escriptori
- **T6** - Reutilització del codi objecte: gestió de biblioteques
- **T7** - Aspectes bàsics i desplegament d'aplicacions Web
- **T8** - Accés a BBDD mitjançant un model d'objectes
- **T9** - Realització de presentacions efectives

- 1 **Avaluació Contínua:** Pràctica individual. Es realitzaran 3 pràctiques individuals. Pràctica 1: 2,5%, Pràctica 2: 7,5%, Pràctica 3: 10% **Puntuació: 20%.**
- 2 **Avaluació Contínua:** Test escriptori. Es realitzarà un test per avaluar els coneixements dels alumnes de forma individual a meitat de curs. Nota mínima necessària: 4. **Puntuació: 30%.**
- 3 **Avaluació Contínua:** Pràctica en grup. Es realitzarà una pràctica en grup sobre una aplicació Web de forma col·laborativa el lliurament de la qual serà a final de curs. A més s'ha de realitzar una exposició d'aquesta pràctica **Puntuació: 30%.**
- 4 **Avaluació Contínua:** Test web. Es realitzarà un test sobre la part web en la data oficial assignada per l'escola politècnica al juny. Nota mínima necessària: 4. **Puntuació: 20%.**

Avaluació sense superar avaluació contínua

- **Atenció!** Al juliol, els alumnes que no superin les activitats d'avaluació contínua hauran de realitzar un examen la puntuació màxima del qual serà 50%.
- Les notes obtingudes en les pràctiques durant el curs, no són recuperables. Es manté la seva nota per calcular la nota mitjana al juliol.
- **Para més detall, al campus virtual veure documento Criteris d'Avaluació Fada".**

Pla d'aprenentatge (I)

Sem.	Ud.	Desc. trab. pres.	Desc. trab. no pres.
01	1	Introducció a l'assignatura. Seminari d'introducció al llenguatge de programació.	-
02	2	Control de versions	Autopràctica guiada per comprendre l'entorn de programació.
03	2	Control de versions	Pràctica 1
04	3	Programació dirigida per esdeveniments i execució diferida de codi	Pràctica 1
05	4	Interfícies gràfiques d'usuari	Pràctica 2
06	5	Accés a BBDD des d'aplicacions d'escriptori	Pràctica 3
07	6	Biblioteques.	Pràctica 3

Sem.	Ud.	Desc. trab. pres.	Desc. trab. no pres.
08	7	Introducció a C# i aplicacions Web	Pràctica en grup
09	8	Model de capes	Pràctica en grup
Prova objectiva (test)			
10	7	Capa d'interfície aplicacions Web	Pràctica en grup
11	7	Capa d'interfície aplicacions Web (II)	Pràctica en grup
12	8	Accés a BBDD manera connectada	Pràctica en grup
13	8,9	Accés a BBDD manera desconnectada. Presentacions efectives	Pràctica en grup
14	7	Aspectes avançats en el desenvolupament d'aplicacions Web	Pràctica en grup. Exposició oral.
15	1-9	Repàs i dubtes	Correcció pràctica en grup
Total		60	90

- ❶ **Pràctiques individuals** Llenguatge Vala.
- ❷ **Pràctica en grup** Llenguatge C# (amb Asp.net).

Introducció a Vala

- Vala és un nou llenguatge de programació: Vala
- Empra les funcionalitats proporcionades per Glib y GObject
- El compilador de Vala genera codi 'C', el qual és compilat per un compilador de **Llenguatge C**.
- És un llenguatge similar a Java i C#, més semblat a aquest últim.

Característiques de Vala

- ❶ POO (classes, classes abstractes, mixin interfícies, polymorphism)
- ❷ Espais de noms (namespaces)
- ❸ Delegats
- ❹ Propietats
- ❺ Senyals
- ❻ Notificacions automàtiques de modificació de propietats
- ❼ Foreach
- ❽ Expressions Lambda / Clausures
- ❾ Inferència de tipus de variables locals
- ❿ Tipus Generics
- ⓫ Tipus No-nuls
- ⓬ Gestió automàtica de memòria dinàmica (automatic reference counting)
- ⓭ Destructors deterministes (RAII)
- ⓮ Excepcions (checked exceptions)
- ⓯ Mètodes Asíncrons (coroutines)
- ⓰ Precondicions i postcondicions (programació per contracte)
- ⓱ Run-time type information
- ⓲ Constructors amb nom
- ⓳ Cadenes Verbatim
- ⓴ Trossejat de arrays i cadenes
- ⓵ Compilació condicional
- ⓶ Sintaxi similar a C#
- ⓷ Compatibilitat a nivell de ABI amb C.

- Selecció: if, else, switch, case, default
- Iteració: do, while, for, foreach, in
- Salt: break, continue, return
- Excepcions: try, catch, finally, throw
- Sincronització: lock
- Declaració de tipus: class, interface, struct, enum, delegate, errordomain
- Modificadors de tipus: const, weak, unowned, dynamic
- Modificadors: abstract, virtual, override, sealed, extern, static, async, inline, new
- Modificadors d'accés: public, private, protected, internal
- Paràmetres de mètodes: out, ref
- Programació per contracte: throws, requires, ensures
- Espais de noms: namespace, using
- Operadors: as, is, in, new, delete, sizeof, typeof
- Accés: this, base
- Literals: null, true, false
- Propietats: get, set, construct, default, value
- Blocs constructors: construct, static construct, class construct
- Unes altres: void, var, yield, global, owned

- Aritmètics: +, -, *, /, %
- Bit a bit: ~, &, |, ^, <<, >>
- Relacionals: <, >, <=, >=
- Igualtat: ==, !=
- Lògics: !, &&, ||
- Assignació: =, +=, -=, /=, %=, &=, |=, ^=, <<=, >>=
- Increment, Decremento: ++, --
- Capdavanters: &, *, -->, delete
- Condicionals: ?
- Comparació amb null: ??
- Concatenació de cadenes: +
- Invocació de mètodes: ()
- Accés a membres: .
- Índex: []
- Trossejat: [:]
- Lambda: =>
- Casting: (Type), (!), as
- Comprovació de tipus en temps d'execució: is
- Transferència de propietat: (owned)
- Qualificador d'àlies d'espais de noms: :: (currently only with global)
- Unes altres: new, sizeof, typeof, in

Hola Món en Vala

Compilació

```
1 class Demo.HelloWorld : GLib.Object {
2     public static int main(string[] args) {
3         stdout.printf("Hello, World\n");
4         return 0;
5     }
6 }
```

- \$ valac compiler.vala --pkg libvala
- \$ valac source1.vala source2.vala -o myprogram
- \$ valac hello.vala -C -H hello.h

```

1  int a = 6, b = 7;
2  string s = @"$a * $b = $(a * b)"; // => "6 * 7 = 42"

4  string greeting = "hello, world";
   string s1 = greeting[7:12]; // => "world"
6  string s2 = greeting[-4:-2]; // => "or"

8  bool b = bool.parse("false"); // => false
   int i = int.parse("-52"); // => -52
10 double d = double.parse("6.67428E-11"); // => 6.67428E-11
   string s1 = true.to_string(); // => "true"
12 string s2 = 21.to_string(); // => "21"

14 if ("ere" in "Able was I ere I saw Elba.") ...

```

```

1  stdout.printf("Hello, world\n");
   stdout.printf("%d %g %s\n", 42, 3.1415, "Vala");
3  string input = stdin.read_line();
   int number = int.parse(stdin.read_line());

```

- També disposem de la sortida d'error estàndard representada por "stderr".
- Podem mostrar informació en ella amb "printf" así: `stderr.printf('...')`;

Arrays

```

1  int[] a = new int[10];
2  int[] b = { 2, 4, 6, 8 };
   int[] c = b[1:3]; // => { 4, 6 }
4  int al = a.length;

6  int[,] c = new int[3,4];
   int[,] d = {{2, 4, 6, 8},
7             {3, 5, 7, 9},
8             {1, 3, 5, 7}};
10 d[2,3] = 42;
   int d0l = d.length[0];
12 int[] e = {}; e += 12; e += 5; e += 37;

```

Classes

```

1  /* defining a class */
   class Track : GLib.Object { // subclassing 'GLib.Object' */
3     public double mass; // a public field */
   public double name { get; set; } // a public property */
5     private bool terminated = false; // a private field */
   public void terminate() { // a public method */
7         terminated = true;
9     }
}

```

```
1  int i = 10;
   float j = (float) i;
3
   var p = new Person(); // same as: Person p = new Person();
   var s = "hello"; // same as: string s = "hello";
5   var l = new List<int>(); // same as: List<int> l = new List<int>();
   var i = 10; // same as: int i = 10;
7
9   MyFoo<string, MyBar<string, int>> foo = new MyFoo<string, MyBar<
   string, int>>();
   // Compara amb...
11  var foo = new MyFoo<string, MyBar<string, int>>();
```

```
1  stdout.printf("Hello, %s!\n", name ?? "unknown person");
```

Foreach

```
1  foreach (int a in int_array) { stdout.printf("%d\n", a); }
```

Comprovació automàtica de valors nuls

```
1  string? method_name(string? text, Foo? foo, Bar bar) {
   // ...
3  }
5  Object o1 = new Object(); // not nullable
   Object? o2 = new Object(); // nullable
7
   o1 = o2; // Prohibit
9  o1 = (!) o2; // Permes amb el cast senar—null explicito: operador !
```

```

1  delegate void DelegateType(int a);

3  void f1(int a) {
4      stdout.printf("%d\n", a);
5  }

7  void f2(DelegateType d, int a) {
8      d(a);          // Calling a delegate
9  }

11 void main() {
12     f2(f1, 5);      // Passing a method as delegate argument to another
13                     method
14 }
```

```

delegate void PrintIntFunc(int a);

4  void main() {
5      PrintIntFunc p1 = (a) => { stdout.printf("%d\n", a); };
6      p1(10);
7      // Curly braces are optional if the body contains only one statement
8      PrintIntFunc p2 = (a) => stdout.printf("%d\n", a);
9      p2(20);
10 }
```

```

1  namespace Hada {
2      int n;
3  }

4

6  using Hada;
7  n = 3;      // O tambien...
8  Hada.n = 3;
```

public	Sense restriccions d'accés
private	Accés limitat des de dins de la definició de la classe o estructura. Aquest és l'accés per defecte si no es diu gens.
protected	Accés limitat des de dins de la definició de la classe o estructura i des de qualsevol classe que derivi d'ella.
internal	Accés limitat des de classes definides en el mateix paquet

```

1  public class Button : Object {
2      public Button() {
3      }
4
5      public Button.with_label(string label) {
6      }
7
8      public Button.from_stock(string stock_id) {
9      }
10 }
11
12 class Demo : Object {
13     ~Demo() {
14         stdout.printf("in destructor");
15     }
16 }

```

```

1  public class Test : GLib.Object {
2      public signal void sig_1(int a);
3
4      public static int main(string[] args) {
5          Test t1 = new Test();
6
7          t1.sig_1.connect( (t, a) => {stdout.printf("%d\n", a);} );
8
9          t1.sig_1(5);
10
11         return 0;
12     }
13 }

```

Propietats

```

1  class Person : Object {
2      private int _age = 32; // underscore prefix to avoid name clash
3      with property
4
5      /* Property */
6      public int age {
7          get { return _age; }
8          set { _age = value; }
9      }
10
11     // O mes resumit...
12     class Person : Object {
13         /* Property with standard getter and setter and default value */
14         public int age { get; set; default = 32; }
15         ...
16         // De solament lectura
17         public int age2 { get; private set; default = 32; }
18     }
19
20     Person alice = new Person();
21     alice.notify["age"].connect (
22         (s, p) => {stdout.printf("age has changed\n");}
23     );

```

Classes abstractes

```

1  public abstract class Animal : Object {
2      public void eat() {
3          stdout.printf("*chomp chomp*\n");
4      }
5
6      public abstract void say_hello();
7  }
8
9  public class Tiger : Animal {
10     public override void say_hello() {
11         stdout.printf("*roar*\n");
12     }
13 }
14
15 public class Duck : Animal {
16     public override void say_hello() {
17         stdout.printf("*quack*\n");
18     }
19 }

```



```

1 public interface ITest : GLib.Object {
2     public abstract int data_1 { get; set; }
3     public abstract void method_1();
4 }
5 ....
6 public class Test1 : GLib.Object, ITest {
7     public int data_1 { get; set; }
8     public void method_1() {
9     }
10 }

```

```

1 class SuperClass : GLib.Object {
2     public virtual void method_1() {
3         stdout.printf("SuperClass.method_1()\n");
4     }
5 }
6
7 class SubClass : SuperClass {
8     public override void method_1() {
9         stdout.printf("SubClass.method_1()\n");
10    }
11 }

```

```

1 bool b = object is SomeTypeName;
2 Type type = object.get_type();
3 stdout.printf("%s\n", type.name());
4
5 Type type = typeof(Foo);
6 Foo foo = (Foo) Object.new(type);

```

```

1 Button b = widget as Button;
2 // Lo anterior equivale a....
3 Button b = (widget is Button) ? (Button) widget : null;

```

```

1 public class Wrapper<G> : GLib.Object {
2     private G data;
3
4     public void set_data(G data) {
5         this.data = data;
6     }
7
8     public G get_data() {
9         return this.data;
10    }
11 }
12
13 var wrapper = new Wrapper<string>();
14 wrapper.set_data("test");
15 var data = wrapper.get_data();

```

```

1 double method_name(int x, double d)
2     requires (x > 0 && x < 10)
3     requires (d >= 0.0 && d <= 1.0)
4     ensures (result >= 0.0 && result <= 10.0)
5 {
6     return d * x;
7 }

```

On **result** és una variable especial que representa el resultat.

Excepcions

```

1 errordomain IOError {
2     FILE_NOT_FOUND
3 }
4
5 void my_method() throws IOError {
6     // ...
7     if (something_went_wrong) {
8         throw new IOError.FILE_NOT_FOUND(
9             "Requested file could not be found.");
10    }
11 }
12
13 try {
14     my_method();
15 } catch (IOError e) {
16     stdout.printf("Error: %s\n", e.message);
17 }
18
19 IOChannel channel;
20 try {
21     channel = new IOChannel.file("/tmp/my_lock", "w");
22 } catch (FileError e) {
23     if (e.is FileError.EXIST) {
24         throw e;
25     }
26     GLib.error("", e.message);
27 }

```

Adreça dels paràmetres

```

1 void method_1(int a, out int b, ref int c) { ... }
2 void method_2(Object o, out Object p, ref Object q) { ... }
3
4 int a = 1;
5 int b;
6 int c = 3;
7 method_1(a, out b, ref c);
8
9 Object o = new Object();
10 Object p;
11 Object q = new Object();
12 method_2(o, out p, ref q);
13
14 // Una implementacion de method_1
15 void method_1(int a, out int b, ref int c) {
16     b = a + c;
17     c = 3;
18 }

```

- Es defineixen fora del nucli del llenguatge en una biblioteca.
- Aquesta biblioteca es diu `Gee` o libgee.
- Les col·leccions disponibles en Gee són:
 - 1 Lists: Col·leccions ordenades de items accessibles per un índex numèric.
 - 2 Sets: Col·leccions no ordenades.
 - 3 Maps: Col·leccions no ordenades de items accessibles per un índex numèric o d'un altre tipus.
- Algunes classes de Gee:
 - `ArrayList<G>`
 - `HashMap<K,V>`
 - `HashSet<G>`

```

1  using Gee;
2
3  void main () {
4      var list = new ArrayList<int> ();
5      list.add (1);
6      list.add (2);
7      list.add (5);
8      list.add (4);
9      list.insert (2, 3);
10     list.remove_at (3);
11     foreach (int i in list) {
12         stdout.printf ("%d\n", i);
13     }
14     list[2] = 10; // same as list.set (2, 10)
15     stdout.printf ("%d\n", list[2]); // same as list.get (2)
16 }

```

Compilar y ejecutar:

```

1 $ valac --pkg gee-1.0 gee-list.vala
2 $ ./gee-list

```

Suport multi-thread

<https://live.gnome.org/Vala/Tutorial>

```

1  void* thread_func() {
2      stdout.printf ("Thread running.\n");
3      return null;
4  }
5
6  int main(string[] args) {
7      if (!Thread.supported()) {
8          stderr.printf ("Cannot run without threads.\n");
9          return 1;
10     }
11
12     try {
13         Thread.create(thread_func, false);
14     } catch (ThreadError e) {
15         return 1;
16     }
17
18     return 0;
19 }
20
21 // Aquest tipus de codigo s'ha de compilar asi:
22 > valac --thread thread_sample.vala

```

Enllaços d'interès

- [Vala para programadors en C#](#)
- [Vala para programadors en Java](#)
- [La gestió de memòria dinàmica en Vala](#)
- Llista de [biblioteques](#) preparades per ser usades des de Vala
- Preguntes freqüents en Vala: [FAQ](#)
- Un tutorial en vídeo que mostra el senzill que és crear una aplicació en vala amb interfície gràfic: [video-tutorial](#)
- [Exemples senzills](#), [exemples de nivell mitjà](#), [exemples amb cadenes](#), [exemples amb senyals i callbacks](#), [exemples amb propietats](#)