Tema: I
Presentació.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant

Contingut

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.

Guia Docent

- **Campus Virtual** $\rightarrow$ **Recursos d'aprenentatge** $\rightarrow$ **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.

Continguts

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 objecto: 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

Avaluació

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 sober 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 parteix web en la data oficial assignada per l'escola politénica al juny. Nota mínima necessària: 4. **Puntuació: 20%**.

lsi Department of Software and Computing Systems

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 Çriteris 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	Autopráctica 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

Pla d'aprenentatge (II)

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**

Llenguatges de pràctiques

1. **Pràctiques individuals** Llenguatge Vala.
2. **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

1. POO (classes, classes abstractes, mixin interfícies, polymorphism)
2. Espais de noms (namespaces)
3. Delegats
4. Propietats
5. Senyals
6. Notificacions automaticas de modificació de propietats
7. Foreach
8. Expressions Lambda / Clausures
9. Inferència de tipus de variables locals
10. Tipus Genericos
11. Tipus No-nuls
12. Gestion automàtica de memòria dinàmica (automatic reference counting)
13. Destructors deterministes (RAII)
14. Excepcions (checked exceptions)
15. Mètodes Asíncrons (coroutines)
16. Precondicions i postcondiciones (programació per contracte)
17. Run-time type information
18. Constructors amb nom
19. Cadenas Verbatim
20. Trossejat de arrays i cadenes
21. Compilacion condicional
22. Sintaxi similar a C#
23. Compatibilitat a nivell de ABI amb C.

Paraules reservades

- Selecció: `if`, `else`, `switch`, `casi`, `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`, `signal`, `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`

Operadors

- 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: .
- Indice: [ ]
- Trossejat: [:]
- Lambda: =>
- Casting: (Type), (!), as
- Comprovació de tipus en temps d'execució: is
- Transferència de propietat: (owned)
- Cualificador d'àlies d'espais de noms: :: (currently only with global)
- Uns altres: new, sizeof, typeof, in

Hola Món en Vala

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

Compilació

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

Cadenas

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

```
    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.") ...
```

Cadenas

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

```
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`" així: `stderr.printf(``...'');`.

Arrays

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

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

Classes

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

```
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   }
```

Conversió i inferència de tipus

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

```
1   int   i = 10;
    float j = (float) i;
3
    var p = new Person();       // same as: Person p = new Person();
5   var s = "hello";            // same as: string s = "hello";
    var l = new List<int>();    // same as: List<int> l = new List<int>();
7   var i = 10;                 // same as: int i = 10;

9   MyFoo<string, MyBar<string, int>> foo = new MyFoo<string, MyBar<
        string, int>>();
    // Compara amb...
11  var foo = new MyFoo<string, MyBar<string, int>>();
```

Operador ??

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

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

Foreach

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

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

Comprovació automàtica de valors nuls

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

```
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 !
```

Delegats

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

```
1   delegate void DelegateType(int a);

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

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

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

Clausures

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

```
delegate void PrintIntFunc(int a);

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

Espais de noms

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

```
namespace Hada {
    int n;
}

using Hada;
n = 3;   // O tambien...
Hada.n = 3;
```

Visibilitat

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

public	Sense restriccions d'accés
private	Accés limitat des de dins de la definicón de la classe o estructura. Aquest és l'accés per defecte si no es diu gens.
protected	Accés limitat des de dins de la definicón 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

Constructors/Destructors

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

```
1   public class Button : Object {
      public Button() {
3     }

5     public Button.with_label(string label) {
      }
7
      public Button.from_stock(string stock_id) {
9     }
    }
11
    class Demo : Object {
13    ~Demo() {
        stdout.printf("in destructor");
15    }
    }
```

Senyals

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

```
public class Test : GLib.Object {
  public signal void sig_1(int a);

  public static int main(string[] args) {
    Test t1 = new Test();

    t1.sig_1.connect( (t, a) => {stdout.printf("%d\n", a);} );

    t1.sig_1(5);

    return 0;
  }
}
```

Propietats

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

```
1   class Person : Object {
      private int _age = 32;  // underscore prefix to avoid name clash
          with property
3
      /* Property */
5     public int age {
        get { return _age; }
7       set { _age = value; }
      }
9   }

11  // O mes resumit...
    class Person : Object {
13    /* Property with standard getter and setter and default value */
      public int age { get; set; default = 32; }
15    ...
      // De solament lectura
17    public int age2 { get; private set; default = 32; }
    }
19
    Person alice = new Person;
21  alice.notify["age"].connect (
        (s, p) => {stdout.printf("age has changed\n");}
23    );
```

Classes abstractes

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

```
1   public abstract class Animal : Object {
      public void eat() {
3       stdout.printf("*chomp chomp*\n");
      }
5
      public abstract void say_hello();
7   }

9   public class Tiger : Animal {
      public override void say_hello() {
11      stdout.printf("*roar*\n");
      }
13  }

15  public class Duck : Animal {
      public override void say_hello() {
17      stdout.printf("*quack*\n");
      }
19  }
```

Interfícies

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

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

Enllaç dinàmic de mètodes

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

```
class SuperClass : GLib.Object {
  public virtual void method_1() {
    stdout.printf("SuperClass.method_1()\n");
  }
}

class SubClass : SuperClass {
  public override void method_1() {
    stdout.printf("SubClass.method_1()\n");
  }
}
```

RTTI

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

```
1   bool b = object is SomeTypeName;
    Type type = object.get_type();
3   stdout.printf("%s\n", type.name());

5   Type type = typeof(Foo);
    Foo foo = (Foo) Object.new(type);
```

Conversions de tipus dinàmiques

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

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

Classes genèriques

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

```
1   public class Wrapper<G> : GLib.Object {
      private G data;
3
      public void set_data(G data) {
5       this.data = data;
      }
7
      public G get_data() {
9       return this.data;
      }
11  }

13  var wrapper = new Wrapper<string>();
    wrapper.set_data("test");
15  var data = wrapper.get_data();
```

Programació per contracte

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

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

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

Excepcions

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

```
errordomain IOError {
  FILE_NOT_FOUND
}

void my_method() throws IOError {
  // ...
  if (something_went_wrong) {
    throw new IOError.FILE_NOT_FOUND(
                          "Requested file could not be found.");
  }
}
...
try {
  my_method();
} catch (IOError e) {
  stdout.printf("Error: %s\n", e.message);
}
...
IOChannel channel;
try {
  channel = new IOChannel.file("/tmp/my_lock", "w");
} catch (FileError e) {
  if(e is FileError.EXIST) {
    throw e;
  }
  GLib.error("", e.message);
}
```

Adreça dels paràmetres

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

```
1   void method_1(int a, out int b, ref int c) { ... }
    void method_2(Object o, out Object p, ref Object q) { ... }
3
    int a = 1;
5   int b;
    int c = 3;
7   method_1(a, out b, ref c);

9   Object o = new Object();
    Object p;
11  Object q = new Object();
    method_2(o, out p, ref q);
13
    // Una implementacion de method_1
15  void method_1(int a, out int b, ref int c) {
      b = a + c;
17    c = 3;
    }
```

Col·leccions (I)

- 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.
- Algunas clases de Gee:
  - ArrayList<G>
  - HashMap<K,V>
  - HashSet<G>

Col·leccions (II)

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

```
using Gee;

void main () {
  var list = new ArrayList<int> ();
  list.add (1);
  list.add (2);
  list.add (5);
  list.add (4);
  list.insert (2, 3);
  list.remove_at (3);
  foreach (int i in list) {
    stdout.printf ("%d\n", i);
  }
  list[2] = 10;                          // same as list.set (2, 10)
  stdout.printf ("%d\n", list[2]);       // same as list.get (2)
}
```

Compilar y ejecutar:

```
$ valac --pkg gee-1.0 gee-list.vala
$ ./gee-list
```

Suporti multi-thread

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

```
void* thread_func() {
  stdout.printf("Thread running.\n");
  return null;
}

int main(string[] args) {
  if (!Thread.supported()) {
    stderr.printf("Cannot run without threads.\n");
    return 1;
  }

  try {
    Thread.create(thread_func, false);
  } catch (ThreadError e) {
    return 1;
  }

  return 0;
}

// Aquest tipus de codigo s'ha de compilar asi:
> 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