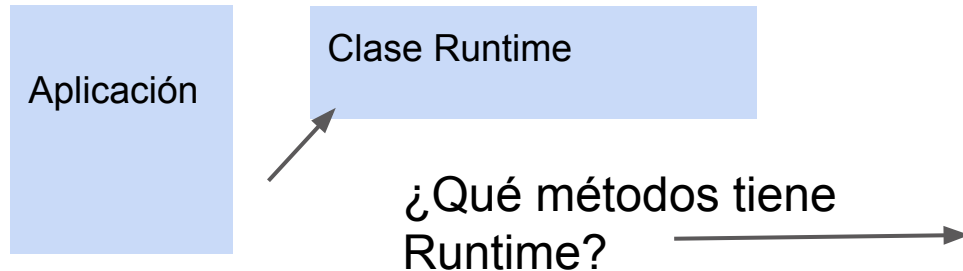

Programación multiproceso

Creación de procesos en
Java

Process y Runtime

Java dispone de varias clases para la gestión de procesos.

Cada aplicación dispone de una instancia Runtime que representa entorno de la aplicación.



Process y Runtime

Métodos importantes:

- *static Runtime* *getRuntime()* devuelve objeto Runtime asociado a aplicación en curso.
 - `Process exec(String comando)` ejecuta la orden especificada en comando en un proceso separado. Devuelve objeto process que se puede utilizar para controlar interacción del programa java con el nuevo proceso.
-

Ejemplo: ejecutar notepad

```
public class Ejemplo1{  
    public static void main(String[] args) {  
        Runtime r=Runtime.getRuntime();  
        String comando="notepad"; //en linux sustituir por comando linux  
        Process p;  
  
        try{  
            p=r.exec(comando);  
        } catch(Exception e) {  
            System.out.println ("Error en "+comando);  
            e.printStackTrace();  
        }  
    }  
}
```

Diagram annotations:

- Arrow from `Runtime r=Runtime.getRuntime();` to "obtengo objeto Runtime"
- Arrow from `Process p;` to "Vble tipo proceso"
- Arrow from `p=r.exec(comando);` to "Ejecuto comando"

```
C:\pss>javac Ejemplo1.java  
C:\pss>java Ejemplo1
```

}//Ejemplo1

Ejemplo: ejecutar notepad

¿Comandos Windows o Linux no tienen ejecutable?

→ Utilizamos CMD /C(wind) ls(linux)

windows → Ejemplo string comando="CMD /C /DIR"

```
C:\pss>javac Ejemplo1dir.java  
C:\pss>java Ejemplo1dir
```

→ No obtengo salida
¿POR QUÉ?

Process


Salida del comando se redirige hacia nuestro programa en Java.


Clase Process

Posee el método **getInputStream()** que nos permite leer el stream de salida del proceso, es decir lo que el comando escribió en la consola de salida. Vamos a verlo en código →

Process . Guardar stream salida

`p=r.exec(comando);`  ejecuto comando

`InputStream is=p.getInputStream();`  guardo stream salida en vle is de tipo InputStream

`BufferedReader br=new BufferedReader(new InputStreamReader (is));`
 guardo en vlble br de tipo BufferedReader para luego sacar linea a linea

Process. Mostrar stream salida

```
String linea;  
while ((linea=br.readLine())!=null)    //lee una línea  
    System.out.println(linea);
```



Lee línea a línea de br y va mostrando
con System.out.println(linea)

Process. Mostrar stream salida

Método **waitFor()** hace que el proceso actual espere hasta que el subprocesso representado por el **Process** finalice.

Devuelve 0 si ha finalizado correctamente.

Ejemplo2

```
import java.io.*;

public class Ejemplo2{

    public static void main(String[] args) {

        Runtime r=Runtime.getRuntime();
        String comando="cmd /c dir";
        Process p=null;

        try{

            p=r.exec(comando);
            InputStream is=p.getInputStream();
            BufferedReader br=new BufferedReader(new InputStreamReader (is));
            String linea;
            while ((linea=br.readLine())!=null){//lee una linea
                System.out.println(linea);
            }
            br.close();
        } catch(Exception e) {
            System.out.println ("Error en "+comando);
            e.printStackTrace();
        }

        //comprobación de error 0 bien - 1 mal
        int exitVal;
        try {
            exitVal=p.waitFor();
            System.out.println("Valor de salida "+exitVal);
        } catch(InterruptedException e){
            e.printStackTrace();
        }

    }

}

//Ejemplo2
```

```
C:\pss>javac Ejemplo2.java
```

```
C:\pss>java Ejemplo2
```

Volume in drive C has no label.
Volume Serial Number is 067B-F907

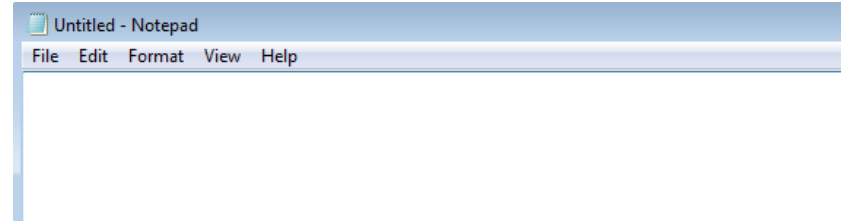
Directory of C:\pss

```
10/10/2013 11:05 <DIR> .
10/10/2013 11:05 <DIR> ..
10/10/2013 10:42 928 Ejemplo1.class
10/10/2013 10:42 282 Ejemplo1.java
10/10/2013 10:58 937 Ejemplo1dir.class
10/10/2013 10:58 302 Ejemplo1dir.java
10/10/2013 11:06 1.277 Ejemplo2.class
10/10/2013 11:06 528 Ejemplo2.java
        6 File(s)          4.254 bytes
        2 Dir(s)  32.386.023.424 bytes free
```


Actividad Casibash

Realiza un programa Java usando Runtime y Process que reciba desde la línea de comandos un nombre de comando y lo ejecute.

```
C:\pss>java act_14  
Introduce comando  
notepad  
  
C:\pss>
```



Process. getErrorStream()

Process posee el método **getErrorStream()**.
Nos permite leer posibles errores se produzcan al lanzar el proceso.

Edita el ejemplo2.java y modifica el comando por “cmd /c dir” o “ls” para provocar error.

La salida es

```
C:\pss>javac Ejemplo2.java
C:\pss>java Ejemplo2
Valor de salida 1
```

Process. getErrorStream()

Si añadimos el siguiente código a ejemplo2:

```
try {
```

```
    InputStream er=p.getErrorStream();
```

```
    BufferedReader brer=new BufferedReader(new InputStreamReader(er));
```

```
    String liner=null;
```

```
    while ((liner=brer.readLine())!=null)
```

```
        System.out.println ("ERROR >"+ liner);
```

```
    }catch (IOException ioe){ ioe.printStackTrace();    }
```

stream de errores de process



guardo en buffer y leo
línea a línea

Process. getErrorStream()

Obtenemos la siguiente salida:

```
C:\pss>java Ejemplo2
Valor de salida 1
ERROR >'dirr' is not recognized as an internal or external command,
ERROR >operable program or batch file.
```

Process. FileOutputStream y PrintWriter

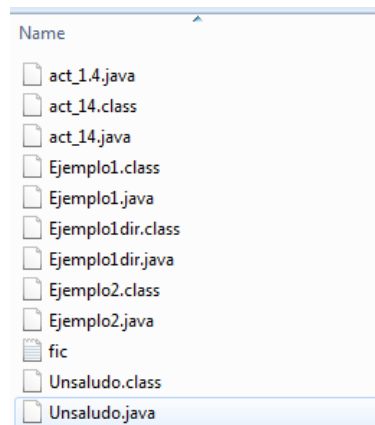
Creo programa Java pinta 5 veces saludo se le introduce por línea comandos

```
public class Unsaludo {  
    public static void main(String[] args){  
        if (args.length<1){  
            System.out.println ("Se necesita saludo...");  
            System.exit(1);  
        }  
        for (int i=0;i<5;i++){  
            System.out.println(i+1 + ". "+ args[0]);  
        }  
    }  
}  
//fin main  
}  
//unsaludo
```

Compilamos y ejecutamos..

```
C:\pss>javac Unsaludo.java  
C:\pss>java Unsaludo "HOLA MUNDO" > fic.txt
```

Se crea fichero
con salida



Process. FileOutputStream y PrintWriter

Cambiamos en ejemplo2 el comando y escribimos:

String comando="java Unsaludo \"Hola Mundo!!\">ejem.txt";

```
C:\pss>javac Ejemplo2.java
```

```
C:\pss>java Ejemplo2
1.Hola Mundo!!>ejem.txt
2.Hola Mundo!!>ejem.txt
3.Hola Mundo!!>ejem.txt
4.Hola Mundo!!>ejem.txt
5.Hola Mundo!!>ejem.txt
Valor de salida 0
```

Compilamos y ejecutamos

No genera fichero salida
¿Por qué ?



Process. FileOutputStream y PrintWriter

exec() no actúa como intérprete de comandos o shell, sólo ejecuta programa.

Si queremos redirigir a un fichero tendremos que añadirlo por programación mediante clases:

- FileOutputStream
 - PrintWriter
-

Ejemplo3. FileOutputStream y PrintWriter

Código 1/3

```
import java.io.*;

public class Ejemplo3 {

    public static void main(String[] args) {

        Runtime r=Runtime.getRuntime(); //objeto runtime
        String comando="java Unsaludo \"Hola Mundo!!\" "; //comando a ejecutar y params
        Process p=null; //inicializo vble process

        //control params entrada
        if (args.length<1){
            System.out.println("Se necesita nombre de fichero..");
            System.exit(1);
        } //fin control parametros
    }
}
```

Ejemplo3. FileOutputStream y PrintWriter

```
import java.io.*;

public class Ejemplo3 {

    public static void main(String[] args) {

        Runtime r=Runtime.getRuntime(); //objeto runtime

        String comando="java Unsaludo \"Hola Mundo!!\" "; //comando a ejecutar y params

        Process p=null; //inicializo vble process


        //control params entrada
        if (args.length<1){

            System.out.println("Se necesita nombre de fichero..");

            System.exit(1);

        }//fin control parametros
```

Ejemplo3. FileOutputStream y PrintWriter

Código 2/3

creo objeto
fileoutputstream y
printwriter

imprimo en
fichero

```
try {  
    //fichero al que se enviará la salida del programa Unsaludo  
    FileOutputStream fos=new FileOutputStream(args[0]);  
    PrintWriter pw=new PrintWriter(fos);  
  
    //ejecuto comando  
    p=r.exec(comando);  
  
    //creo buffer  
    InputStream is=p.getInputStream();  
    BufferedReader br=new BufferedReader(new InputStreamReader (is));  
    String linea;  
    //leo linea a linea e imprimo en el fichero  
    while ((linea=br.readLine())!=null)//lee una linea  
    {  
        System.out.println("Inserto en "+args[0]+" > "+linea);  
        pw.println(linea); //la inserto en el fichero  
    }//fin bucle lineas  
    br.close();  
    pw.close();  
}  
catch (Exception e) {e.printStackTrace();}
```

Ejemplo3. FileOutputStream y PrintWriter

```
try {  
  
    //fichero al que se enviará la salida del programa Unsaludo  
    FileOutputStream fos=new FileOutputStream(args[0]);  
    PrintWriter pw=new PrintWriter(fos);  
  
    //ejecuto comando  
    p=r.exec(comando);  
  
    //creo buffer  
    InputStream is=p.getInputStream();  
    BufferedReader br=new BufferedReader(new InputStreamReader (is));  
    String linea;  
    //leo linea a linea e imprimo en el fichero  
    while ((linea=br.readLine())!=null)//lee una linea  
    {  
        System.out.println("Inserto en "+args[0]+" > "+linea);  
        pw.println(linea); //la inserto en el fichero  
    }//fin bucle lineas  
    br.close();  
    pw.close();  
}  
    catch (Exception e) {e.printStackTrace();}
```

Ejemplo3. FileOutputStream y PrintWriter

Código 3/3

```
//comprobación de error 0 bien - 1 mal
int exitVal;
try {
    exitVal=p.waitFor();
    System.out.println("Valor de salida "+exitVal);
} catch (InterruptedException e) {
    e.printStackTrace();
}

} //fin main
} //fin Ejemplo3
```

Ejemplo3. FileOutputStream y PrintWriter

```
//comprobación de error 0 bien - 1 mal
    int exitVal;
    try {
        exitVal=p.waitFor();
        System.out.println("Valor de salida "+exitVal);
    } catch (InterruptedException e){
        e.printStackTrace();
    }

    }//fin main
} //fin Ejemplo3
```


Actividad Servisvhost

Realiza un programa que generará proceso que guarda en un fichero de nombre SVHOST.TXT los servicios que están ejecutando bajo el proceso svhost.exe

Svchost.exe es un proceso de su PC que aloja, o contiene, otros servicios individuales que usa Windows para ejecutar diversas funciones. Por ejemplo, Windows Defender usa un servicio alojado por un proceso de svchost.exe.

Process. getErrorStream()

La clas Process posee el método
getErrorStream()

Permite obtener un stream para poder leer los
posibles errores producidos al lanzar al
proceso.

Antes sólo obteniamos valor salida 1 o 0 si
funcionaba correctamente o no..

¿Qué podemos obtener con getStreamError()?

Process. getOutputStream

Process posee método `getOutputStream()` que permite escribir en el stream de entrada del proceso.

Es decir, pasarle los valores que nos solicitará el proceso.

Ejemplo ejecuta Date

```
C:\pss>date
The current date is: 11/10/2013
Enter the new date: <dd-mm-yy> 10/10/2013_
```

Ejemplo5. Windows

Vamos a implementar el Ejemplo5 que ejecutará date y le pasará los valores 10/10/2013.

```
import java.io.*;
public class Ejemplo5 {
    public static void main(String[] args){
        Runtime r=Runtime.getRuntime();
        String comando="CMD /C DATE"; //comando a ejecutar
        Process p=null;
```


Primera parte igual que todos los ejemplos sólo cambia el comando

Ejemplo5

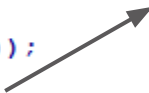
```
try{
    p=r.exec(comando);
    //escritura - envio entrada a date
    OutputStream os=p.getOutputStream();
    os.write("10-10-13".getBytes());
    os.flush(); //vacio buffer

    //lectura - obtiene la salida de date
    InputStream is=p.getInputStream();
    BufferedReader br=new BufferedReader(new InputStreamReader (is));
    String linea;
    while ((linea=br.readLine())!=null)//lee una linea
        System.out.println(linea);
    br.close();
} catch(Exception e) {
    System.out.println ("Error en "+comando);
    e.printStackTrace();
}
```

Añadimos escritura, para el envío de datos al proceso p



lectura igual en ejemplos anteriores



Ejemplo5

```
//comprobación de error 0 bien - 1 mal
int exitVal;
try {
    exitVal=p.waitFor();
    System.out.println("Valor de salida "+exitVal);
} catch (InterruptedException e) {
    e.printStackTrace();
}

} //fin main
```

comprobación error igual en otros ejemplos

Ejemplo5. Código nuevo

//escritura - envio entrada a date

```
OutputStream os=p.getOutputStream();
```

```
os.write("10-10-13".getBytes());
```

```
os.flush(); //vacio buffer
```

Ejemplo 5 Linux

Modificamos comando Windows para linux

Ejecutamos comando passwd

String comando="passwd"; //comando a ejecutar

Process p=null;

try{

 p=r.exec(comando);

 //escritura - envio entrada

 OutputStream os=p.getOutputStream();

 os.write("david123\n".getBytes());

 os.flush(); //vacio buffer

os.write("simarro123\n".getBytes());

os.flush(); //vacio buffer

os.write("simarro123\n".getBytes());

os.flush(); //vacio buffer

Process. getOutputStream

Realizamos programa Java lea una cadena de la entrada estándar EjemploLectura.java

```
import java.io.*;
public class EjemploLectura{
    public static void main (String[] args){
        InputStreamReader in=new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader (in);
        String texto;

        try{
            System.out.println("Introduce una cadena...");
            texto=br.readLine();
            System.out.println("Cadena escrita: "+texto);
            in.close();
        }catch (Exception e) {e.printStackTrace();}

    } //fin main
} //fin ejemplo lectura
```

Process. getOutputStream

Utilizando el método `getOutputStream()` podemos enviar datos a la entrada estándar
EjemploLectura.

Actividad

Realiza modificaciones en Ejemplo5 para que realice lo anterior y le pase la cadena “Hola Simarro”. Guarda como Ejemplo6.java

La salida del programa debe ser

```
C:\pss>javac Ejemplo6.java  
C:\pss>java Ejemplo6  
Introduce una cadena...  
Cadena escrita: Hola Simarro  
Valor de salida 0
```

Actividad Sumados

Escribe un programa en Java que lea dos números de la entrada estándar y visualice su suma. Haz otro programa Java para ejecutar el anterior sin introducir los números por consola.

Salida del programa esperada

```
C:\pss>javac Act_162.java
C:\pss>java Act_162
Introduce un numero
Introduce otro numero
la suma es: 3
Valor de salida 0
```