

INSTITUTO TECNOLÓGICO DE OAXACA

INGENIERÍA EN SISTEMAS
COMPUTACIONALES

TÓPICOS AVANZADOS DE
PROGRAMACIÓN

SU

LIC. ADELINA CRUZ NIETO

UNIDAD III

INVESTIGACIÓN II

ANAYA BAUTISTA GUADALUPE
BERENICE

14/07/18

A) Creación de componentes (visuales y no visuales) definidos por el usuario.

¿Cómo se crea un componente en NetBeans?

El formato de ficheros de "Archivos Java" permite empaquetar varios ficheros en un sólo archivo. Típicamente un fichero JAR contendrá los ficheros de clases y los recursos auxiliares asociados con los programas y aplicaciones. Estos recursos auxiliares podrían incluir, por ejemplo, ficheros de imagen y sonido que sean utilizados por un programa.

PASOS PARA CREAR UN COMPONENTE:

1. Crear un proyecto en NetBeans.
2. Crea un archivo .java (JPanel Form).

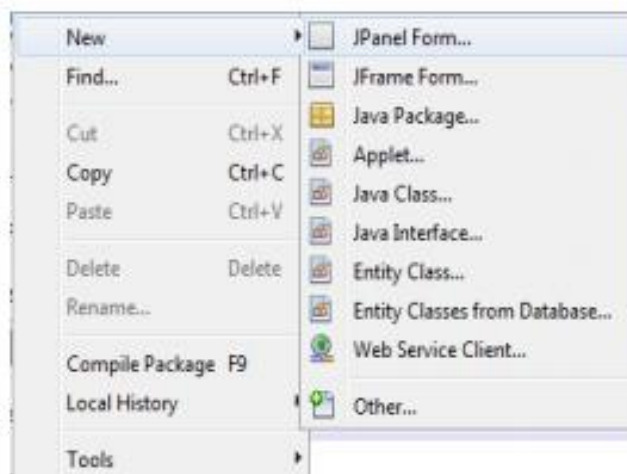


Fig. 3.4.1 Creación JPanel Form.

3. Ya creado el JPanel Form agregar los componentes y código correspondientes para la funcionalidad del programa.
4. Crear el .jar de la aplicación.



Fig. 3.4.2 Creación de extensión JAR.

Run ---- Build Proyect

5. Ya creado el archivo .jar del JPanel, se agrega en a paleta de componentes.

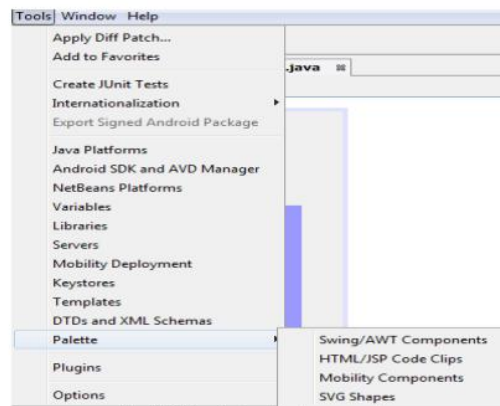


Fig. 3.4.3 Creación de paleta de componentes.

6. Se crea una nueva categoría(New Category) y después se agrega el .jar (Add from JAR...)

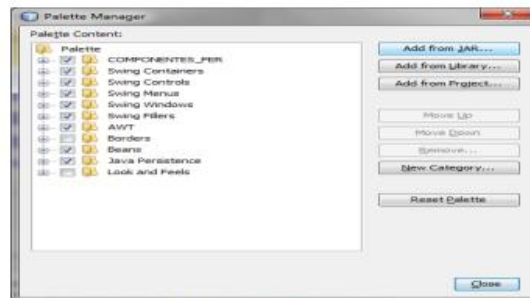


Fig. 3.4.4 Selección de componentes.

7. Se busca el .jar y se carga:

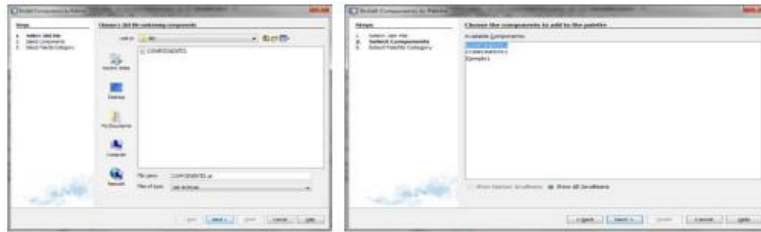


Fig. 3.4.5 Búsqueda de componente.

8. Se selecciona la carpeta donde se desea cargar el .jar..

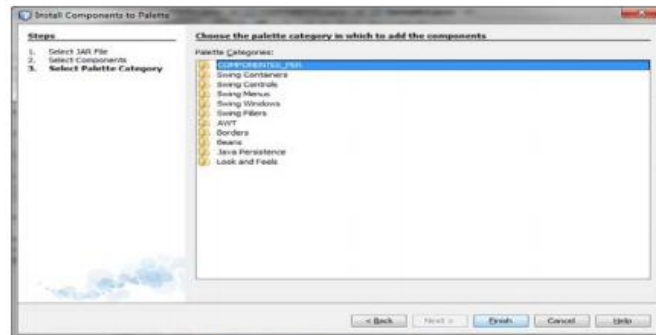


Fig. 3.4.6 Selección de categoría de componentes.

9. Después de cargar ya se puede utilizar el componente y debe aparecer en la paleta de componentes en la sección cargada.



B.1) Creación y uso de paquetes/librerías definidas por el usuario

Los paquetes son el mecanismo por el que Java permite agrupar clases, interfaces, excepciones y constantes. De esta forma, se agrupan conjuntos de estructuras de datos y de clases con algún tipo de relación en común. Con la idea de mantener la reutilización y facilidad de uso de los paquetes desarrollados es conveniente que las clases e interfaces contenidas en los mismos tengan cierta relación funcional.

De esta manera los desarrolladores ya tendrán una idea de lo que están buscando y fácilmente sabrán qué pueden encontrar dentro de un paquete. Creación de un paquete:

1. Declaración Para declarar un paquete se utiliza la sentencia `package` seguida del nombre del paquete que estemos creando:

```
package NombrePaquete;
```

La estructura que ha de seguir un fichero fuente en Java es:

- Una única sentencia de paquete (opcional).
- Las sentencias de importación deseadas (opcional).
- La declaración de una (y sólo una) clase pública (`public`).
- Las clases privadas del paquete (opcional).

Por lo tanto la sentencia de declaración de paquete ha de ser la primera en un archivo fuente Java.

2. Nomenclatura. Para que los nombres de paquete puedan ser fácilmente reutilizados en toda una compañía o incluso en todo el mundo es conveniente darles nombres únicos. Esto puede ser una tarea realmente tediosa dentro de una gran empresa, y absolutamente imposible dentro de la comunidad de Internet.

3. Subpaquetes. Cada paquete puede tener a su vez paquetes con contenidos parecidos, de forma que un programador probablemente estará interesado en organizar sus paquetes de forma jerárquica. Para eso se definen los subpaquetes.

```
package paquete.mipaquete;  
  
import paquete2.otropaquete.*;  
  
class Ejemplo1{}
```

Para crear un subpaquete bastará con almacenar el paquete hijo en un directorio Paquete/Subpaquete.

El JDK define una variable de entorno denominada CLASSPATH que gestiona las rutas en las que el JDK busca los subpaquetes. El directorio actual suele estar siempre incluido en la variable de entorno CLASSPATH.

Uso de un paquete. Con el fin de importar paquetes ya desarrollados se utiliza la sentencia import seguida del nombre de paquete o paquetes a importar. Se pueden importar todos los elementos de un paquete o sólo algunos. Para importar todas las clases e interfaces de un paquete se utiliza el metacaracter *: import PaquetePrueba.*;

También existe la posibilidad de que se deseen importar sólo algunas de las clases de un cierto paquete o subpaquete: import Paquete.Subpaquete1.Subpaquete2.Clase1;

Para acceder a los elementos de un paquete, no es necesario importar explícitamente el paquete en que aparecen, sino que basta con referenciar el elemento tras una especificación completa de la ruta de paquetes y subpaquetes en que se encuentra. Paquete.Subpaquetes1.Subpaquete2.Clase_o_Interfaz.elemento

En la API de Java se incluyen un conjunto de paquetes ya desarrollados que se pueden incluir en cualquier aplicación (o applet) Java que se desarrolle. Ámbito de los elementos de un paquete. Al introducir el concepto de paquete, surge la duda de cómo proteger los elementos de una clase, qué visibilidad presentan respecto al resto de elementos del paquete, respecto a los de otros paquetes.

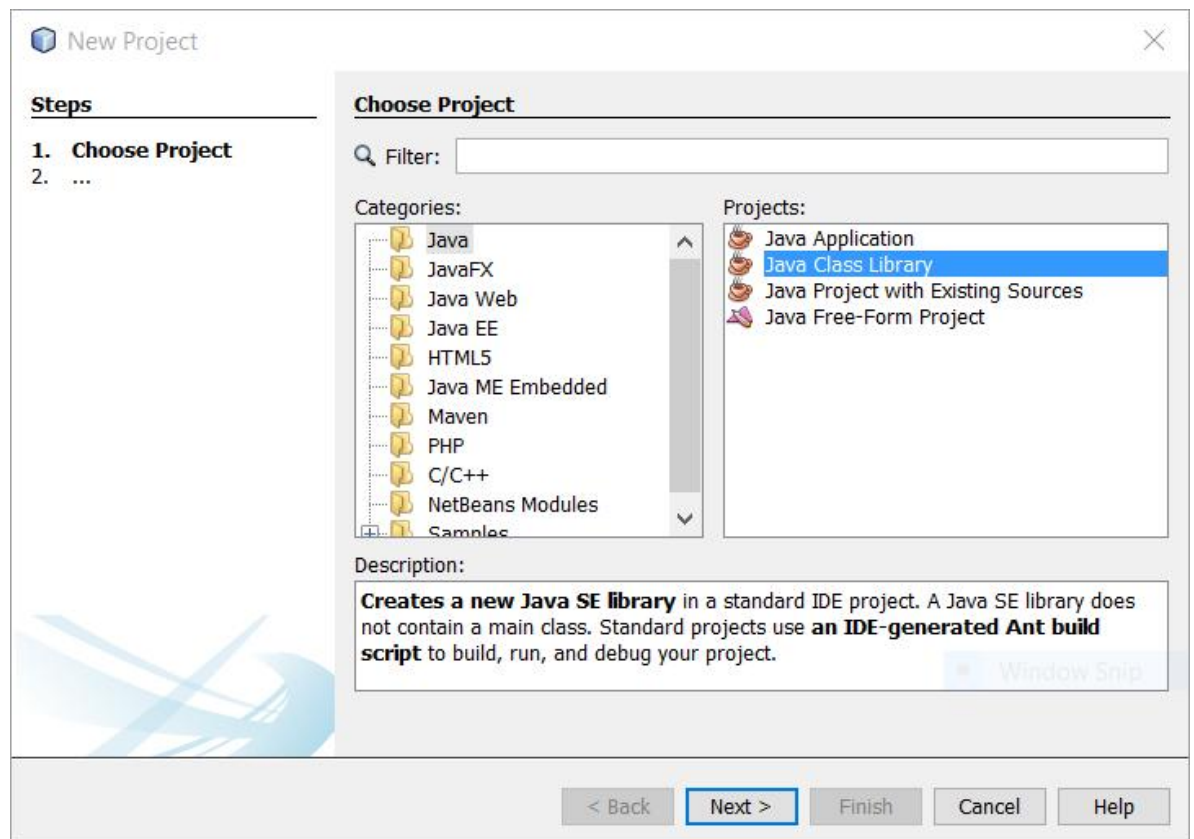
B.2) CREACIÓN Y USO DE PAQUETES / LIBRERÍAS DEFINIDAS POR EL USUARIO.

Las librerías son un conjunto de clases con funciones específicas que ayudan a desarrollar aplicaciones más complejas de una forma sencilla.

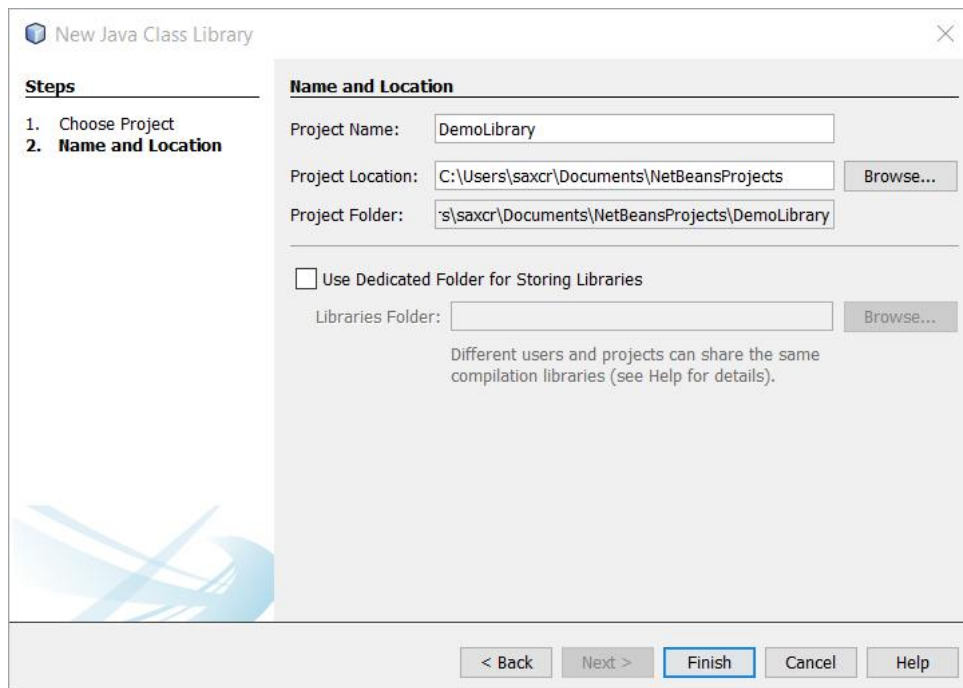
Java cuenta con una extensa lista de librerías disponibles dentro del JDK o desarrolladas por terceros, pero también existe la posibilidad de desarrollar librerías propias para reutilizar de forma más eficiente el código.

Creación de librerías en netbeans.

1. Crear un nuevo proyecto de tipo “Java Class Library”.



2.- En este caso se llama **DemoLibrary**.

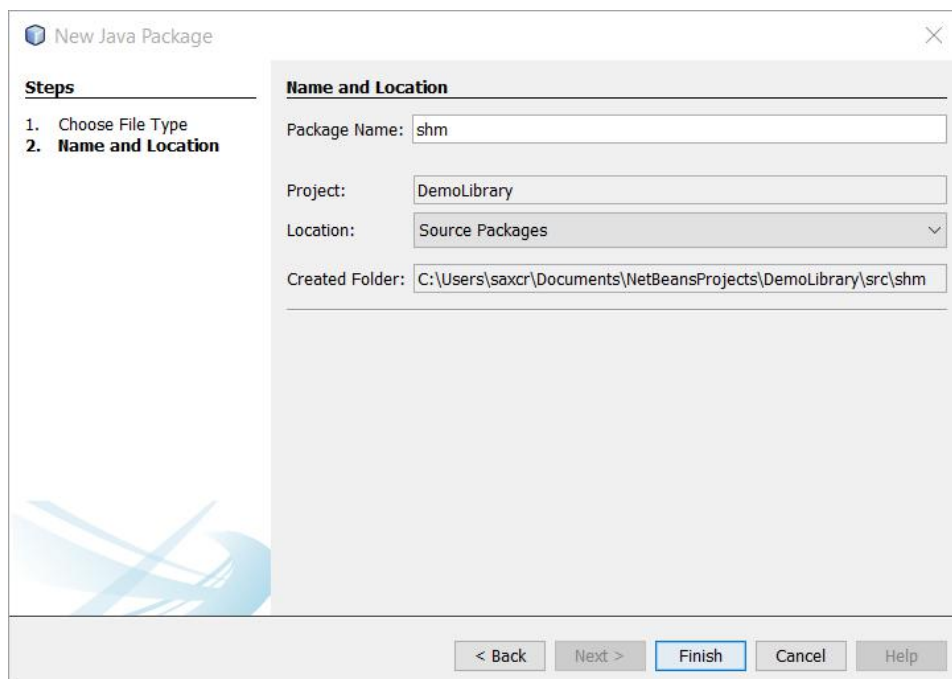


The dialog box is titled "New Java Class Library". On the left, under the "Steps" section, step 2 "Name and Location" is selected. The main area is titled "Name and Location" and contains the following fields:

- Project Name:** DemoLibrary
- Project Location:** C:\Users\saxcr\Documents\NetBeansProjects (with a "Browse..." button)
- Project Folder:** -s\saxcr\Documents\NetBeansProjects\DemoLibrary
- ☐ **Use Dedicated Folder for Storing Libraries**
 - Libraries Folder:** (with a "Browse..." button)
 - Text below: "Different users and projects can share the same compilation libraries (see Help for details)."

At the bottom, there are five buttons: "< Back", "Next >", "Finish" (highlighted with a blue border), "Cancel", and "Help".

3. Después de crear el proyecto se crea un package, en este caso se pueden poner las iniciales o algún otro nombre de referencia.

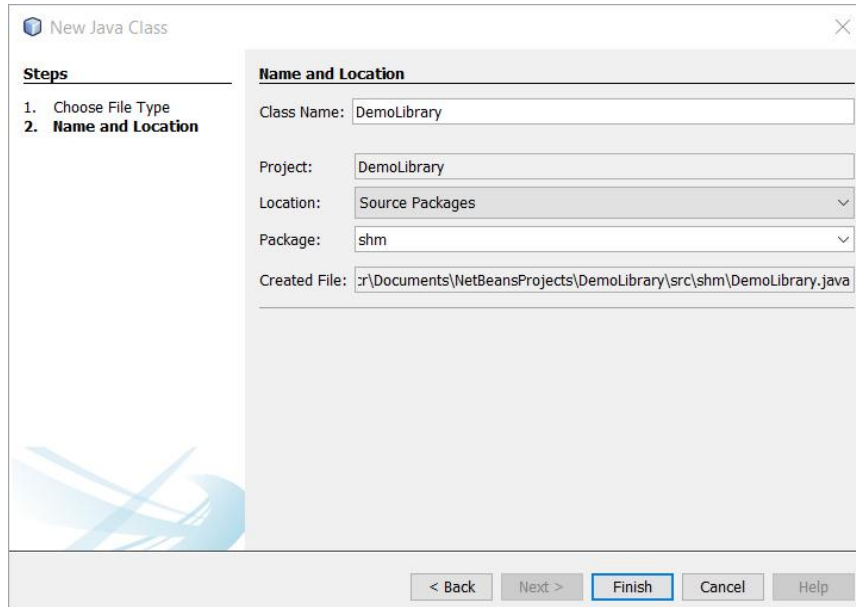


The dialog box is titled "New Java Package". On the left, under the "Steps" section, step 2 "Name and Location" is selected. The main area is titled "Name and Location" and contains the following fields:

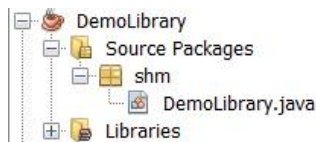
- Package Name:** shm
- Project:** DemoLibrary
- Location:** Source Packages (dropdown menu)
- Created Folder:** C:\Users\saxcr\Documents\NetBeansProjects\DemoLibrary\src\shm

At the bottom, there are five buttons: "< Back", "Next >", "Finish" (highlighted with a blue border), "Cancel", and "Help".

4. Por último se crea la clase donde se almacenarán los métodos que serán reutilizados, para fines prácticos se llamara igual que el proyecto **“DemoLibrary”**.



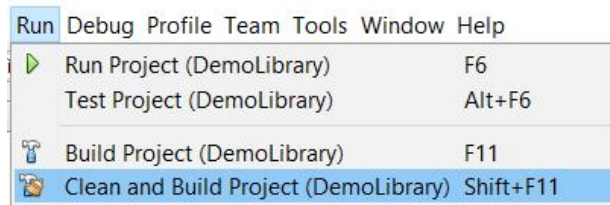
5.-La estructura del proyecto quedará como se muestra en la siguiente imagen.



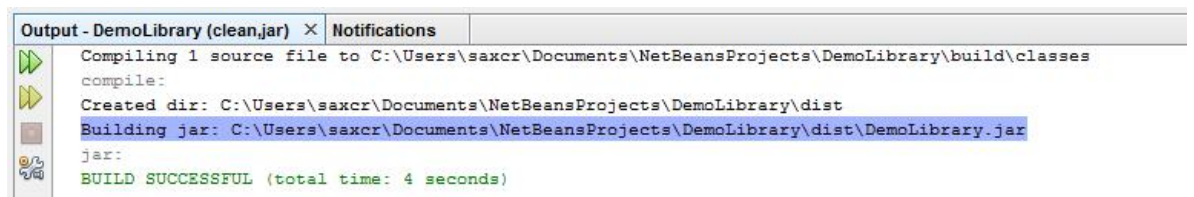
6. Esta clase contiene 4 métodos, para sumar, restar, multiplicar y dividir enteros, métodos muy simples para ejemplificar su aplicación.

```
1  package shm;
2
3  public class DemoLibrary {
4
5      public int suma(int num1, int num2){
6          return num1 + num2;
7      }
8      public int resta(int num1, int num2){
9          return num1 - num2;
10     }
11     public int multiplicacion(int num1, int num2){
12         return num1 * num2;
13     }
14     public int division(int num1, int num2){
15         if(num2!=0)
16             return num1 / num2;
17         else{
18             System.err.println("La división entre 0 es una indeterminación");
19             return 0;
20         }
21     }
22 }
```

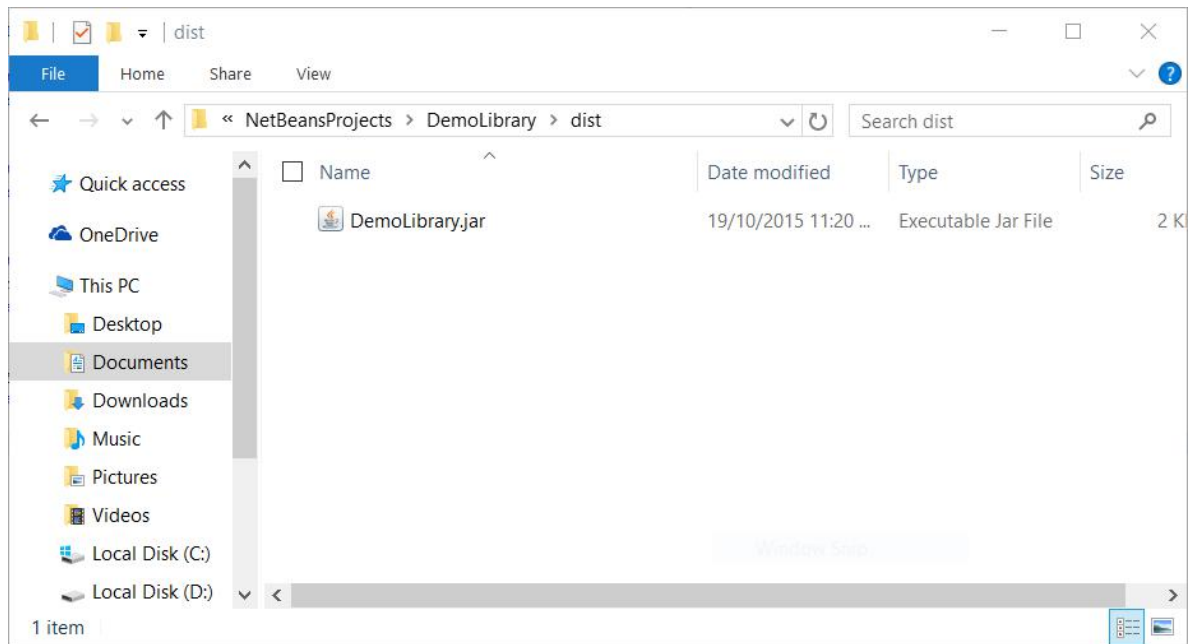
7. Para generar la librería se accede al menú **Run**, y con la opción **Clean and Build Project**, se borran todas las clases compiladas previamente, se vuelven a compilar y se genera un paquete **.jar**, que es el formato en el que se distribuyen los ejecutables de java.



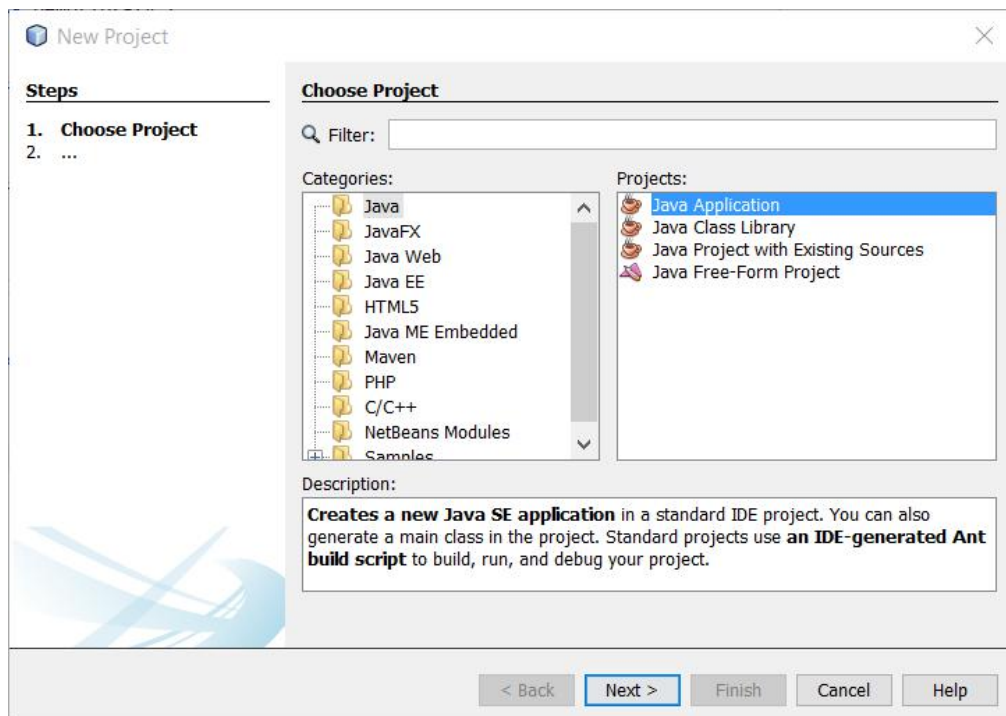
8. Si la compilación es exitosa se mostrara la leyenda **Build SUCCESSFUL**, y además se indica que se creó una carpeta de nombre **dist** y dentro de ella la librería **jar**.



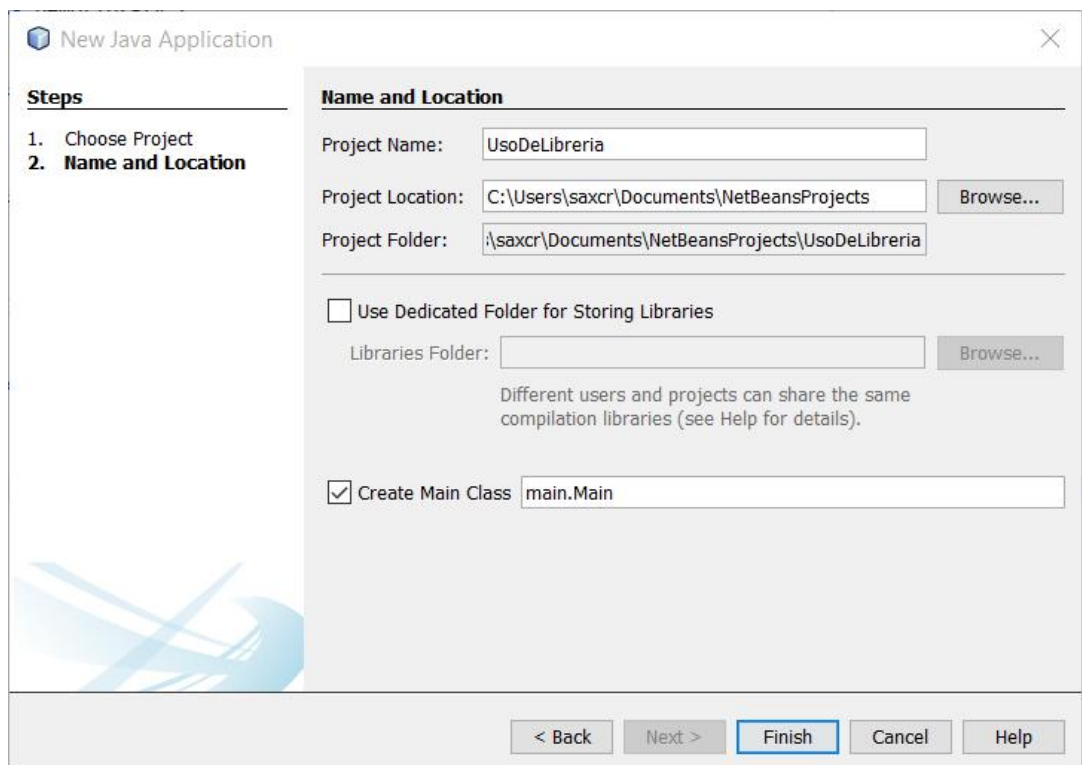
9. Al abrir en el explorador la carpeta **dist**, se puede ver el ejecutable de java, el archivo **DemoLibrary.jar**, este archivo es el que se debe importar en otro proyecto para hacer uso de los métodos creados.



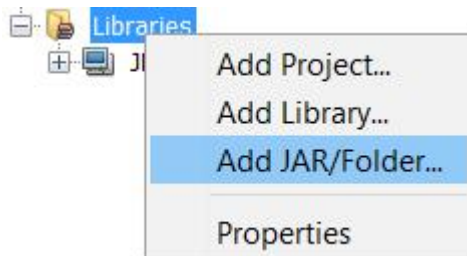
10. Para utilizar la librería creada se genera un nuevo proyecto de tipo **Java Application**.



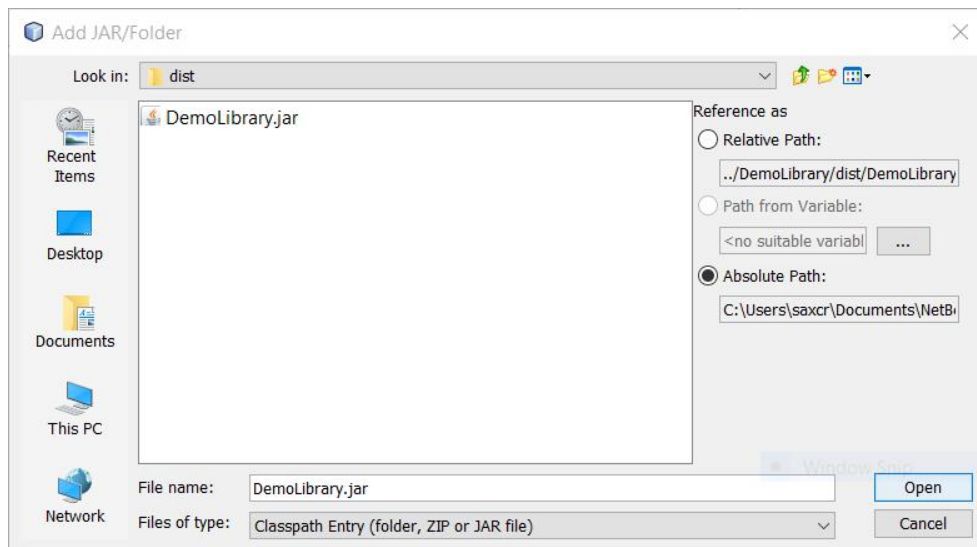
11. En este caso el proyecto lleva por nombre UsoDeLibreria



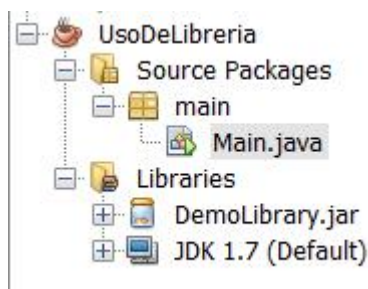
12. Una vez creado el proyecto se importara la librería creada, utilizando la función **Add JAR/Folder** al proyecto.



13. Para esto se utiliza la ubicación de la carpeta **dist**, creada en el proyecto anterior.



14. El esquema del proyecto se ve como en la siguiente imagen.



15. Para utilizar la librería se importa el paquete y la librería, en este caso **import shm.DemoLibrary**, y para su uso se crea una instancia, a través de la cual se accede a los métodos de suma, resta, multiplicación y división.

```
1 package main;
2 import shm.DemoLibrary;
3 public class Main {
4
5     public static void main(String[] args) {
6         DemoLibrary dl = new DemoLibrary();
7         System.out.println("Suma " + dl.suma(10, 15));
8         System.out.println("Resta " + dl.resta(10, 15));
9         System.out.println("Multiplicación " + dl.multiplicacion(10, 15));
10        System.out.println("División " + dl.division(10, 0));
11    }
12 }
```

BIBLIOGRAFIA:

<http://www.tesoem.edu.mx/alumnos/cuadernillos/2013.001.pdf>

<https://netbeans.org/kb/docs/java/quickstart-gui.html>