



Universidad Nacional Autónoma de México

Facultad de Ingeniería



Laboratorio de docencia
Laboratorios de computación
Laboratorios de computación A y
B

Práctica #6 Organización de clases

Profesor: Tista García Edgar

Asignatura: Programación Orientada a Objetos

Grupo: 3

No de Práctica(s): 6

Integrante(s): Félix Flores Paul Jaime

Semestre: 2019-2

Fecha de entrega: 17-03-2019

Observaciones:

CALIFICACIÓN: _____

OBJETIVO.

Organizar adecuadamente las clases según su funcionalidad o propósito bajo un *namespace* o paquete.

DESARROLLO.

Ejercicio 1) Manual de laboratorio.

En este primer ejercicio vimos como importar la biblioteca random con el uso de la librería random y al momento de imprimir la invocamos para obtener el resultado

```
C:\Users\andro>cd Desktop
C:\Users\andro\Desktop>javac Manuala.java
error: Class names, 'Manuala.java', are only accepted when running javac with the -classpath option
1 error
C:\Users\andro\Desktop>javac Manuala.java
C:\Users\andro\Desktop>java Manuala
7
C:\Users\andro\Desktop>
```

Ejercicio 2) Manual de laboratorio

Para que se genere el archivo HolaMundo.class dentro del directorio hola, se compila con -d .

la opción -d y la ruta origen sería el directorio actual, es decir punto (.):

```
Microsoft Windows [Versión 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\andro>cd Desktop

C:\Users\andro\Desktop>dir hola*
El volumen de la unidad C es Nuevo vol
El número de serie del volumen es: 2E82-62BF

Directorio de C:\Users\andro\Desktop

14/03/2019  08:08 a. m.    <DIR>          hola
14/03/2019  08:04 a. m.             223 HolaMundo.java
               1 archivos             223 bytes
               1 dirs  179,961,987,072 bytes libres

C:\Users\andro\Desktop>javac -d . HolaMundo.java

C:\Users\andro\Desktop>dir hola*
El volumen de la unidad C es Nuevo vol
El número de serie del volumen es: 2E82-62BF

Directorio de C:\Users\andro\Desktop

14/03/2019  08:08 a. m.    <DIR>          hola
14/03/2019  08:04 a. m.             223 HolaMundo.java
               1 archivos             223 bytes
               1 dirs  179,957,710,848 bytes libres

C:\Users\andro\Desktop>
```

Ejercicio 3) Manual de laboratorio

Los archivos JAR contienen archivos de clases y recursos de la aplicación. En general un

archivo JAR puede contener:

- Los archivos *.class que se generan a partir de compilar los archivos *.java que

componen la aplicación.

- Los archivos de recursos que necesita la aplicación (Por ejemplo, archivos de configuración, de texto, de sonido, imágenes, etc.)

- Opcionalmente se puede incluir los archivos de código fuente *.java

- Opcionalmente puede existir un archivo de configuración "META-INF/MANIFEST.MF".

Para que el archivo JAR sea ejecutable se debe incluir en el archivo MANIFEST.MF una

línea indicando la clase que contiene el método estático main() que se usará para iniciar la

aplicación. Este archivo se puede generar manualmente con cualquier editor de texto y es

importante destacar que al final de la línea hay que agregar un salto de línea para que

funcione. Si el archivo no se crea manualmente, se puede crear con la herramienta JAR.

```
CA: Símbolo del sistema
El volumen de la unidad C es Nuevo vol
El número de serie del volumen es: 2E82-62BF

Directorio de C:\Users\andro\Desktop\mx\unam\fi\poo

18/03/2019  10:36 a. m.    <DIR>      .
18/03/2019  10:36 a. m.    <DIR>      ..
18/03/2019  10:09 a. m.           470 Manualb.class
18/03/2019  11:10 a. m.           440 MiClase.class
          2 archivos           910 bytes
          2 dirs  177,845,633,024 bytes libres

C:\Users\andro\Desktop>java mx.unam.fi.poo.MiClase
Clase empaquetada

C:\Users\andro\Desktop>jar -cvfe MiJar.jar mx.unam.fi.poo.MiClase mx/unam/fi/poo/*.class
manifiesto agregado
agregando: mx/unam/fi/poo/Manualb.class(entrada = 470) (salida = 336)(desinflado 28%)
agregando: mx/unam/fi/poo/MiClase.class(entrada = 440) (salida = 302)(desinflado 31%)

C:\Users\andro\Desktop>jar tf MiJar.jar
META-INF/
META-INF/MANIFEST.MF
mx/unam/fi/poo/Manualb.class
mx/unam/fi/poo/MiClase.class

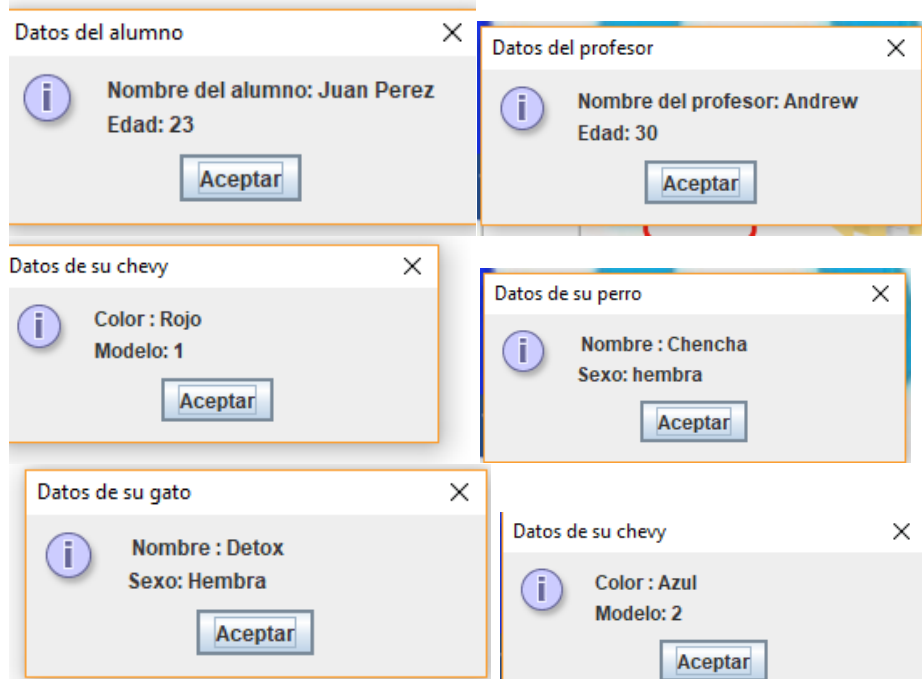
C:\Users\andro\Desktop>java -jar MiJar.jar
Clase empaquetada

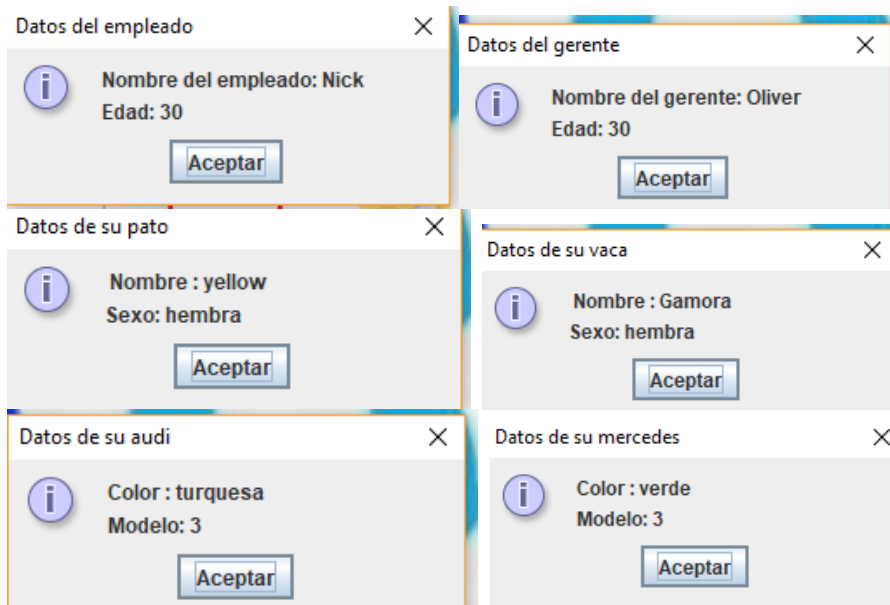
C:\Users\andro\Desktop>
```

Ejercicio 1)

En este ejercicio tuvo muchos problemas en cuanto a información, ya que en el manual de laboratorio ,vienen ejemplos muy sencillos de una clase empaquetada ,pero no dan ejemplos de cómo llamar la herencia entre paquetes o más paquetes por lo cual la información es muy obsoleta para una práctica así, por lo cual no pude usar herencia ,entiendo cómo funcionan los paquetes y como importar paquetes ,pero me confundí a la hora de importar entre paquetes , y en la herencia ya que me decía que los atributos eran públicos ,y al momento de compilar desde consola ,este no me reconocía el paquete ,así que decidí optar por netbeans, creo que pude hacer el cometido de la practica pero de otra forma debido a mi escaso conocimiento y el poco tiempo de buscar información.

Para netbeans decidí crear las instancias en mi clase principal que era practica6 y esta iba a llamar los demás paquetes.





Conclusiones:

En esta práctica pudimos ver la interacción entre paquetes, así como la creación de ellos y cómo importar clases a otros paquetes y crear `archivo.jar` de acuerdo al manual de laboratorio.

Creo que me salieron bien los resultados no en consola por que no pude compilar, no me reconocía la clase, pero lo pude hacer en netbeans, creo que si hubiera tenido un poco más de tiempo hubiera investigado un poco más en youtube acerca de los paquetes, pero con el proyecto y otras materias me es imposible hacer algo bueno en poco tiempo, no quise dejarla en blanco así que intente lo que pude. Pude comprender cómo es la función de los paquetes, sin embargo me quedo la duda de que si se pueden heredar entre paquetes, ya que intente eso y no me salía.