{{ 	**Carátula para entrega de prácticas**
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	M.C. Jorge Rommel Santiago Arce
Asignatura:	Programación Orientada a Objetos
Grupo:	2
No de Práctica(s):	8
Integrante(s):	Cruz Rangel Leonardo Said Ibañez Guzman Osvaldo Maya Ortega Maria Fernanda van der Werff Vargas Pieter Alexander
No. de Equipo de cómputo empleado:	45-48
No. de Lista o Brigada:	7
Semestre:	2020-1
Fecha de entrega:	07/10/2019
Observaciones:	

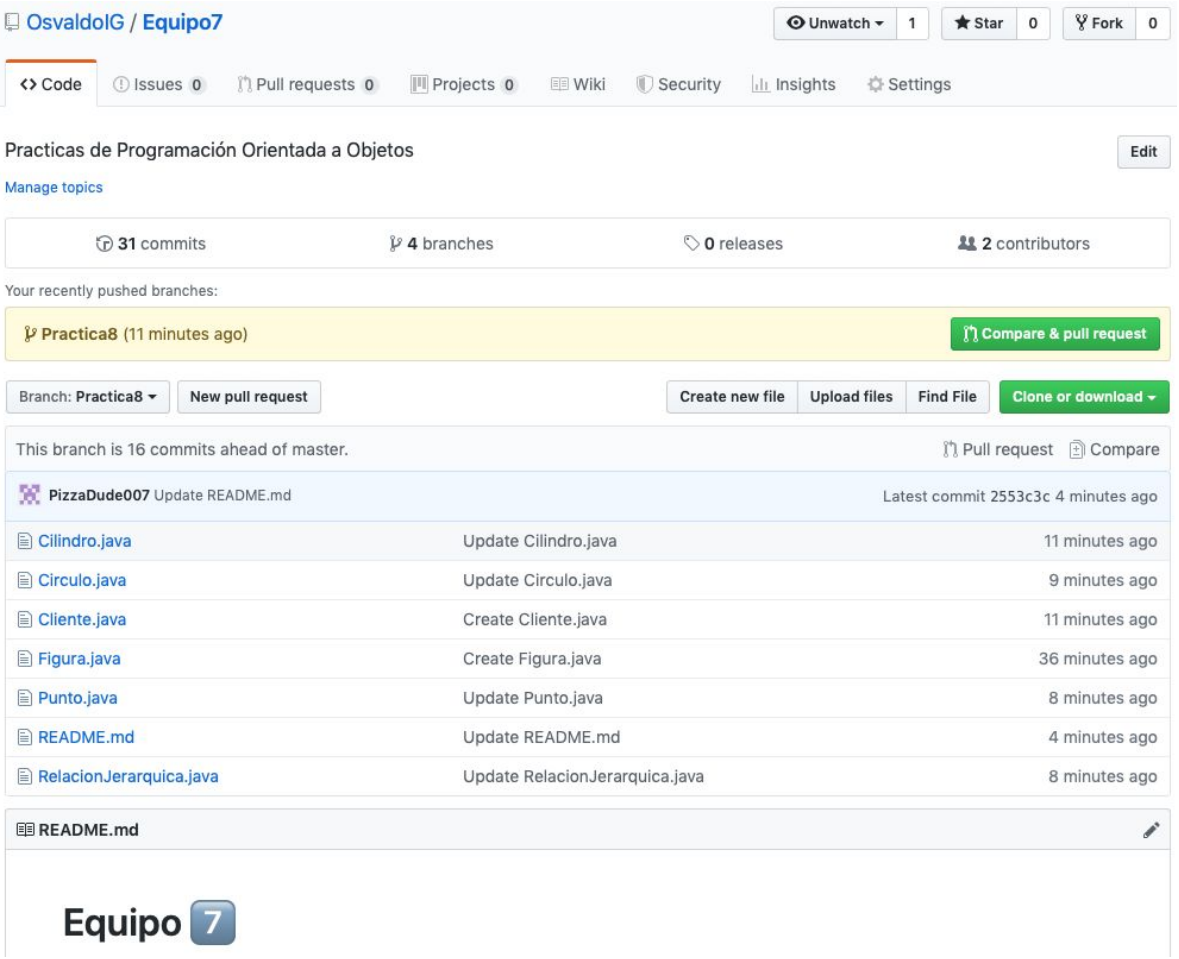
CALIFICACIÓN: _____

PRÁCTICA 8. “Polimorfismo”

Objetivos:

Implementar el concepto de polimorfismo en un lenguaje de programación orientado a objetos.

Actividad 1 - El Repositorio



Nuestro equipo ya contaba con repositorio, pues desde la práctica 6 hemos empezado a trabajar las prácticas de esta manera.

Actividad 2 - Invocación de métodos de superclases desde objetos de subclases

```
[Granada47:Equipo7-Practica8 poo02alu37$ java RelacionJerarquica
Llamado a toString de Punto con referencia a la superficieapuntando al objeto de la superclase:
[30, 50]
Llamado a toString de Cicrulo con referencia a la subclaseapuntando al objeto de la subclase:
Centro = [120, 89], Radio = 2.7
Centro = [120, 89], Radio = 2.7
```

Para la primera impresión tenemos un llamado a *Punto*, simplemente ocupando un

objeto que se ha inicializado en su propio constructor en la superclase, por lo cual solo imprime dos parámetros. Para la segunda impresión, se tiene una impresión de un objeto de *Circulo*, una clase que hereda de *Punto*, en donde se modifica el método *toString*, por lo cual imprime de una manera diferente.

Lo que resulta interesante es la tercera impresión, ya que se está instanciando un objeto de tipo *Punto* con un objeto de tipo *Circulo*, al realizar esto, el nuevo objeto *refPunto*, aunque sea de la clase *Punto* va a implementar los nuevos métodos de la sublclase, ya que se está instanciando con un valor de ella.

Actividad 3 - Referencias a superclases con variables tipo subclase

```
Granada47:Equipo7-Practica8 poo02alu37$ javac RelacionJerarquica.java
RelacionJerarquica.java:5: error: incompatible types: Punto cannot be converted to Circulo
    Circulo circulo = punto;
                      ^
1 error
Granada47:Equipo7-Practica8 poo02alu37$
```

Aunque, las clase circulo hereda de la clase punto, no se pueden igualar. Como observamos en el ejemplo anterior, es posible hacerlo de manera directa de manera inversa ya que toma los métodos de la subclase y los implementa en el objeto de la superclase, pero no funciona si se intenta realizar de manera opuesta.

```
Granada47:Equipo7-Practica8 poo02alu37$ javac RelacionJerarquica.java
Granada47:Equipo7-Practica8 poo02alu37$
```

Si se realiza un conversión, mediante un cast. El código compilara, pero presenta un error en tiempo de ejecución.

```
Granada47:Equipo7-Practica8 poo02alu37$ java RelacionJerarquica
Exception in thread "main" java.lang.ClassCastException: Punto cannot be cast to Circulo
        at RelacionJerarquica.main(RelacionJerarquica.java:5)
```

Esto se debe a que *Circulo*, al ser la subclase tiene los métodos de *Punto*, además de otros adicionales, por lo que sí se intenta realizar una conversión de un objeto de la superclase a la subclase harán falta métodos y atributos, por ejemplo. Por lo tanto no es posible realizar llamado al método *toString*, en este caso, ya que hacen falta campos para imprimir los valores necesarios y se produce una excepción.

Actividad 4 - Herencia y el modificador abstract

Al momento de hacer la herencia entre las clases concretas es necesario que la primera clase que herede use en la totalidad los métodos

De igual forma esto funciona si todas las clases están en un mismo archivo, solo que se deben de omitir la palabra “public” para que pueda compilarse.

Sobreescritura del método obtenerNombre() heredado de la clase abstracta Figura en las clases Punto y Círculo.

```
public String obtenerNombre(){
  return "Punto";
}
```

```
public String obtenerNombre(){
    return "Circulo";
}
```

Modificaciones en la clase Cilindro.

```
// sobrescribir método abstracto obtenerArea para devolver el área de Cilindro
public double obtenerArea(){
    return (2 * Math.PI * obtenerRadio() * obtenerRadio()) + (Math.PI * obtenerRadio() * obtenerAltura());
}

// sobrescribir método abstracto obtenerVolumne para devolver vaolr del cilindro
public double obtenerVolumen(){
    return Math.PI * obtenerRadio() * obtenerRadio() * obtenerAltura();
}

// sobrescribir método abstracto obtenerNombre para devolver "Cilindro"
public String obtenerNombre(){
    return "Cilindro";
}

//sobrescribir toString para devolver representación String del Cilindro

public String toString(){
    return "Radio = "+obtenerRadio()+" Altura = "+obtenerAltura();
}
```

El polimorfismo ocurre en la clase HerenciaAbstacta al momento de declarar el arreglo de figuras y apuntar cada una de sus posiciones a un objeto de las subclases Punto, Círculo y Cilindro, esto se puede hacer porque los métodos de la clase Figura son métodos abstractos, esto quiere decir que la clase Figura puede adoptar la forma de sus clases hijas.

Salida de la clase HerenciaAbstracta, ésta se usa para comprobar nuestras clases de figuras anteriores.

```
Gambia45:Desktop poo02alu11$ javac *.java
Gambia45:Desktop poo02alu11$ java HerenciaAbstracta
Nombre y representacion de cadena:
Punto: [7, 11]
Circulo: Radio = 3.3 Altura = 10.75


Nombre e invocaciones:
Punto
[7, 11]
0.0
0.0


Circulo
Centro = [22, 8], Radio = 3.5
38.48451000647496
0.0


Cilindro
Radio = 3.3 Altura = 10.75
179.8718873812836
367.7783979741231
```

Conclusiones:

Cruz Rangel Leonardo Said:
El polimorfismo aún no lo he comprendido al cien por ciento, creo que es un concepto un poco abstracto y que requiere de muchos ejemplos y ejercicios para poder darle el uso correcto. La práctica me gustó, estuvo entretenida, siempre es importante complementar la teoría con ejercicios prácticos.

Ibañez Guzman Osvaldo:
Se logró ver la importancia y la implementación del Polimorfismo, además de reforzar conceptos importante como son super y las clases abstractas.

Maya Ortega María Fernanda:
En esta ocasión observamos la estrecha relación que existe entre los conceptos polimorfismo y herencia, además de su implementación en jerarquías de clasificación que se dan a través de la herencia.

van der Werff Vargas Pieter Alexander:
Al ocupar herencia en clases, podemos ocupar objetos de una subclase en su superclase, pero no siempre viceversa, por lo que es importante realizar ejemplos, para notar las excepciones que se crean, así como las maneras en que se pueden evitar. Además de que pueden crear conflicto de diferentes maneras, como es el casteo para el objeto de una superclase a una subclase, observamos que aunque compila, crea

conflicto en tiempo de ejecución.