ESTRUCTURA DE DATOS 2015-2

Integrantes:

YESID ALIRIO ESCOBAR GUEVARA

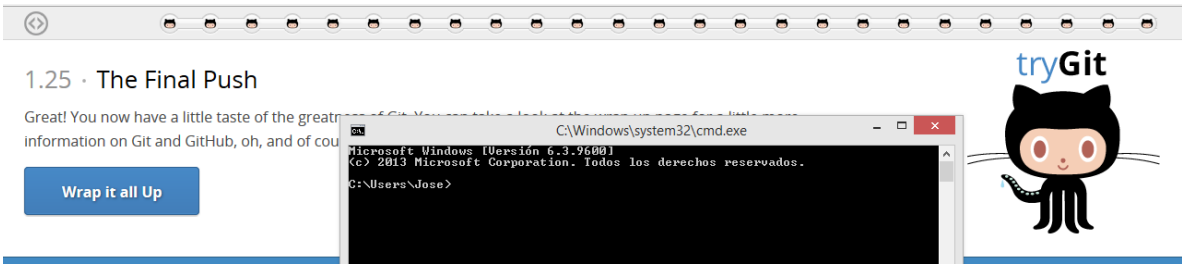
LUIS ESTEBAN CORTES ROMERO

JOSÉ MANUEL RAMÍREZ RESTREPO

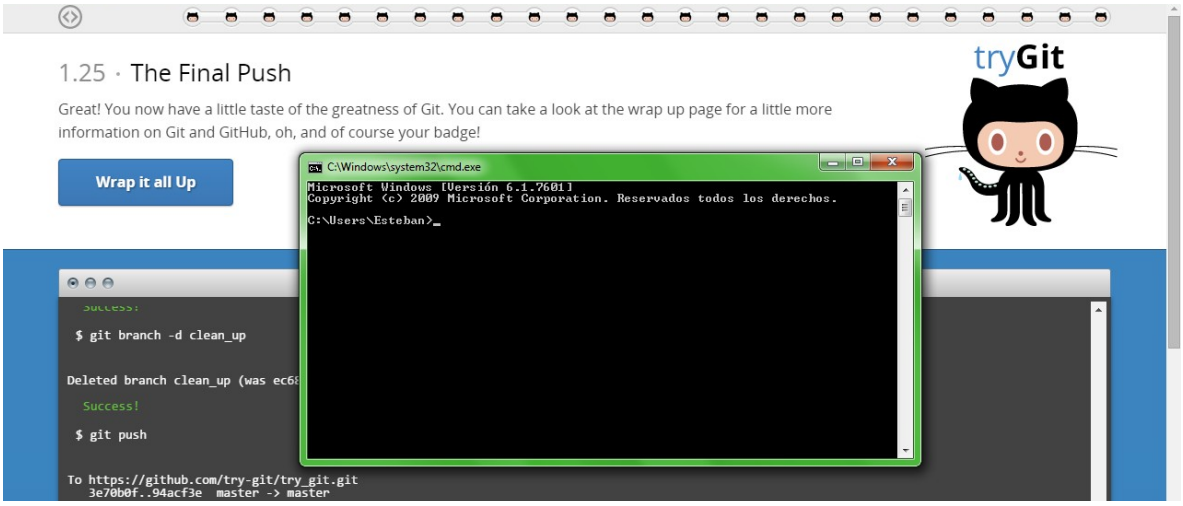
1. Se realizaron lecturas para la comprensión de Git y algunos de los comandos más importantes en Bash.

2. A continuación imágenes prueba del tutorial online

Yesid Escobar



Jose Ramirez



Esteban Cortes

4. a

Abstracción: Es un proceso que se basa en la focalización de los aspectos más relevantes, de un objeto, dejando a un lado todo lo que carezca dicha relevancia.

Por ejemplo si vamos a construir una bicicleta, sabemos que necesitaremos un marco, una cadena, un manubrio, dos llantas, y un sillín, hasta ahí llegaría el proceso de abstracción, es decir, no nos pondremos a pensar cómo funciona cada uno de estos elementos.

Clase: Es un tipo de bolsa descriptiva que contiene datos y operaciones los cuales describen el comportamiento de los elementos u otras bolas que esta contiene, estos elementos son conocidos como objetos.

Por ejemplo podemos tener los objetos, resistor, diodo, y transistor, de ellos podemos decir que todos pertenecen a una clase llamada componentes electrónicos pasivos, y esta clase a su vez podría estar dentro de otra clase llamada componentes electrónicos.

Objeto: Es una entidad provista de un conjunto de propiedades o atributos (datos), de un comportamiento o funcionalidad (métodos) y de sus posibles relaciones con otros objetos.

Por ejemplo tomemos el objeto Automóvil: un automóvil es un objeto bastante pesado que tiene un conjunto de propiedades como su identificación (placa), color, marca, modelo, accesorios, etc. Tiene también un conjunto de funciones como la de desplazarse, detenerse, ponerse en marcha. Podemos cambiarle de color, aumentar o quitar sus accesorios; es decir, podemos modificar sus propiedades.

Atributo: Los atributos son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. Los atributos se guardan en

variables denominadas de instancia, y cada objeto particular puede tener valores distintos para estas variables. (Representan las propiedades que caracterizan la clase.)

Por ejemplo La clase Coche contiene un atributo llamada marca.

Método: Es un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.

Por ejemplo, para un formulario tenemos el método Hide que hará que el formulario se oculte; o el método Show que hará que el formulario se vuelva a mostrar. Hay muchos eventos, propiedades, y demás.

Encapsulamiento: Consiste en hace visible o no atributos o métodos de una clase determinada, esto para otras clases u otros paquetes, existen tres niveles, públicos, privados y protegidos. Se puede entender como una protección a la información y su modificación desde un lugar en específico.

Por ejemplo, cuando alguien ve a otra persona puede saber inmediatamente si es hombre o mujer (propiedad); también puede conocer el color de cabello y ojos. En cambio, jamás sabrá que cantidad de energía exacta tiene o cuantas neuronas se tienen, ni siquiera preguntando ya que ninguna de tus propiedades externas visibles o funciones de comunicación al público permitirán saber esos datos.

Herencia: Es lo que permite el paso o transmisión de métodos y atributos de una clase a otra, además de permitir la existencia de una jerarquía entre las clases.

Por ejemplo si tenemos una clase llamada televisor, y otras llamadas, televisor LCD, televisor plasma, televisor 3D, es claro ver que entre estas últimas existen varias diferencias, pero sin embargo todas tienen funcionalidades y atributos similares, estos atributos y funcionalidades se los podemos asignar al televisor, y decir que los otros tres tienen todo eso, más unas funcionalidades y atributos propios, creando así una jerarquía liderada por la clase televisor, y a su vez haciendo que este HEREDE sus atributos y funcionalidades a las otras tres clases.

Polimorfismo: Es la capacidad de dos o más objetos pertenecientes a diferentes clases para responder a exactamente el mismo mensaje (llamada de método) en diferentes formas específicas de clase.

Por ejemplo Diferentes personas puede realizar un mismo comportamiento como una patada en diferentes formas, ya sea una patada en un partido de fútbol, una patada de un karateca, patear una piedra, etcétera. Es posible decir que responden a exactamente el mismo mensaje en diferentes formas específicas de clase.

Diferencia entre tipo primitivo y de referencia: Las variables de tipos primitivos almacenan directamente un valor que siempre pertenece al rango de ese tipo. Esto significa que al asignar una variable entera a otra variable entera, se copia el valor de la primera en el espacio que ocupa la segunda variable.

Las variables de tipo referencia a objetos en cambio almacenan direcciones y no valores directamente. Una referencia a un objeto es la dirección de un área en memoria destinada a representar ese objeto. El área de memoria se solicita con el operador new. Los primitivos no necesitan métodos y no necesitan ser invocados para su creación, mientras que los de tipo de referencia necesitan métodos y necesitan ser invocados.

Por ejemplo un dato primitivo “int” y un dato de referencia “String”.

Diferencia entre usar private, protected, public, default: Para controlar el acceso a nuestros atributos y métodos se utilizan los modificadores de acceso que no son más que palabras reservadas del lenguaje que se encargarán de controlar desde dónde serán accesibles los miembros de una clase.

- Private (Acceso solo dentro de la clase)
- Protected (Acceso desde la clase y sus hijos "herencia")
- Default (Sin escribir nada, denominado acceso de paquete)
- Public (Acceso público desde cualquier lugar)

MODIFICADOR	CLASE	PACKAGE	SUBCLASE	TODOS
Public	Sí	Sí	Sí	Sí
Protected	Sí	Sí	Sí	No
Default	Sí	Sí	No	No
Private	Sí	No	No	No

Clase Abstracta: Una clase que declara la existencia de métodos pero no la implementación de dichos métodos (o sea, las llaves { } y las sentencias entre ellas), se considera una clase abstracta. Una clase abstracta puede contener métodos no-abstractos pero al menos uno de los métodos debe ser declarado abstracto.

***Interfaz*:** Esta por un lado la llamada Interface que es la parte visible y pública de un método o clase que describe qué hace y cómo usarlo. La documentación de un método en el API de Java vendría siendo su interface. Por otro lado está la llamada Interfaz Gráfica de Usuario o GUI (Graphical User Interface): es el entorno de objetos gráficos disponibles para un usuario en el marco de una aplicación o sistema operativo.

Por ejemplo el sistema operativo MS-Dos se basaba en intérpretes de comando (escritura de instrucciones por consola) pero Windows se basa en una interfaz gráfica de usuario (su entorno de escritorio), Linux en otra y Macintosh en otra.

CÓDIGO

Para la primera parte, se realizó una clase que contenía las propiedades de un rectángulo en base a sus dos lados, ancho y alto como característica extra se añadió una excepción en caso de que los valores del rectángulo sean menores a 0 o mayores a 20. Así mismo se dieron las operaciones necesarias para obtener área y perímetro de un rectángulo y se agregó el rectángulo por defecto como un rectángulo de lados 1.

```
public class Rectangulo {

  private double ancho;
  private double largo;

  public Rectangulo(double ancho, double largo) {
    this.ancho = ancho;
    this.largo = largo;
    if (ancho<1 || largo<1 || ancho>20 || largo>20){
    throw new UnsupportedOperationException("Not supported yet.");
    }
  }

  public static double area(double b, double a) {
    return b * a;
  }

  public static double perimetro(double l1, double l2) {
    return 2 * (l1 + l2);
  }

  Rectangulo() {
    ancho = 1;
    largo = 1;
    }
```

En la misma clase se agregaron getter an setter respectivos y un toString para la clase que imprimar todos los valores del rectángulo.

```
  @Override
  public String toString() {
    return '{' + "ancho=" + ancho + ", largo=" + largo + ", perímetro= " + getPerimetro() + ",
area=  " + getArea() + '}';
  }
```

Así mismo se realizó una clase Main en la cual se ingresa el rectángulo por defecto y 4 rectángulos adicionales con diferentes valores. Todavía no lo hacemos por listas ya que ese paso viene para el siguiente código.

```
public class Main {
  public static void main(String[] args) {
    Rectangulo rectdefault = new Rectangulo();
    System.out.println("Default = " + rectdefault);
    Rectangulo rectangulo1 = new Rectangulo(5, 5);
    System.out.println("Rectangulo 1 = " + rectangulo1);
    Rectangulo rectangulo2 = new Rectangulo(4, 4);
    System.out.println("Rectangulo 2 = " + rectangulo2);
    Rectangulo rectangulo3 = new Rectangulo(3, 5);
    System.out.println("Rectangulo 3 = " + rectangulo3);
    Rectangulo rectangulo4 = new Rectangulo(5, 4);
    System.out.println("Rectangulo 4 = " + rectangulo4);
  }
}
```

Para la segunda parte realizamos diferentes clases con las siguientes figuras: triángulo, círculo y nuevamente rectángulo, cada una con su respectiva área y respectivo perímetro. Para circulo y triangulo realizamos nuevas clase, para rectángulo modificamos la clase previa.

Para círculo:

```
public class circulo implements Figuras {

  private double radio;

  public circulo(double radio) {
    this.radio = radio;
    if (radio <= 0) {
      throw new UnsupportedOperationException("Cambiar el radio por uno mayor a 0");
```

```
    }
  }

  public static double area(double a) {
    return a * a * Math.PI;
  }

  public static double perimetro(double a) {
    return Math.PI * 2 * a;
  }
```

Para triángulo usamos una definición de semiperímetro para relacionarla con área:

```
public class triangulo implements Figuras{

  private double lado1;
  private double lado2;
  private double lado3;

  public triangulo(double lado1, double lado2, double lado3) {
    this.lado1 = lado1;
    this.lado2 = lado2;
    this.lado3 = lado3;
    if (lado1+lado2<lado3 || lado1+lado3<lado2 || lado2+lado3<lado1) {
      throw new UnsupportedOperationException("Ese triángulo no puede ser");
    }
  }

  public static double perimetro(double lado1, double lado2, double lado3) {
    return lado1 + lado2 + lado3;
  }

  public static double semiperimetro(double lado1, double lado2, double lado3) {
    return perimetro(lado1, lado2, lado3) / 2;
  }

  public static double area(double lado1, double lado2, double lado3) {
    double s = semiperimetro(lado1, lado2, lado3);
    return Math.sqrt(s * (s - lado1) * (s - lado2) * (s - lado3));
  }
```

En la clase Figuras colocamos las propiedades que se piden de cada figura:

```
public interface Figuras {

   double getArea();

   double getPerimetro();
}
```

En la clase Main colocamos realizamos una lista con diferentes figuras incluyendo las figuras por defecto. Luego usamos iterator para imprimir cada elemento de la lista con while y hasNext

```
public class Main {

   public static void main(String[] args) {

      ArrayList<Figuras> Arr = new ArrayList<>();

      Arr.add(new Rectangulo());
      Arr.add(new Rectangulo(8, 3));
      Arr.add(new Rectangulo(4, 5));
      Arr.add(new Rectangulo(3, 9));
      Arr.add(new Rectangulo(111.11, 6));
      Arr.add(new triangulo());
      Arr.add(new triangulo(2,2,4));
      Arr.add(new triangulo(3,4,5));
      Arr.add(new triangulo(8,9,5));
      Arr.add(new triangulo(7,7,7));
      Arr.add(new circulo());
      Arr.add(new circulo(2));
      Arr.add(new circulo(20));
      Arr.add(new circulo(5));
      Arr.add(new circulo(10));

      ListIterator<Figuras> iterador = Arr.listIterator();
      while (iterador.hasNext()) {
         System.out.print(iterador.next() + "\n");
      }
   }
}
```

Para finalizar, aclarando que el código se encuentra también en la cuenta de github de https://github.com/yaescobarg/Estructuras-de-Datos-2015-2