Tarea 2
Programación
Universidad Nacional Autónoma de México.
Facultad de Ciencias.

Valeria Ortiz Cervantes

14 de abril del 2019

Índice general

Capítulo 1

Conceptos

1.1. TDA

Un Tipo de Dato Abstracto (TDA) es un conjunto de datos u objetos al cual se le asocian operaciones para facilitar su programación. Se compone de:

1. Estructura de datos: es la estructura de programación que se selecciona para representar las características de la entidad modelada.
2. Funciones de abstracción: son funciones que permiten hacer uso de la estructura de datos, y que esconden los detalles de dicha estructura, permitiendo un mayor nivel de abstracción.

Ejemplo:
Una lista [1,2,3,4,5]

1.2. Interfaz

Una interfaz es un conjunto de métodos abstractos y de constantes cuya funcionalidad es la de determinar el funcionamiento de una clase, es decir, funciona como un molde o como una plantilla. Al ser sus métodos abstractos estos no tienen funcionalidad alguna, sólo se definen su tipo, argumento y tipo de retorno.
Ejemplo:

```
public interface MatematicaVectorial  {
public static double pi = 3.14159;
public double[] sumar(int[][]vectores);
}
```

1.3. Clase abstracta

Una clase que declara la existencia de métodos pero no la implementación de dichos métodos se considera una clase abstracta; puede contener métodos no

abstractos pero al menos uno de los métodos debe ser declarado abstracto.
Ejemplo:

```
public abstract class Figura {
public int area;
abstract public void  calcularArea();
}
```

1.4. Herencia

Es un mecanismo que nos permite definir una clase a partir de la definición de otra ya existente, nos permite compartir automáticamente métodos y datos entre clases, subclases y objetos.
Ejemplo:

```
public class Auto {
public void arrancar() {
bla bla bla
}
}

public class Toyota extends Auto {
public void arrancar() {
super.arrancar();
}
}
```

1.5. Superclase

El concepto de herencia conduce a una estructura jerárquica de clases, dentro de la cual cada clase tiene sólo una clase padre. La clase padre de cualquier clase es conocida como su superclase.
Utilizando el ejemplo anterior la superclase de Toyota sería Auto.

1.6. Encapsulamiento

Es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados. Por lo tanto, el encapsulamiento garantiza la integridad de los datos que contiene un objeto.
Ejemplo:

```
public class MiClase{
private int tipo;
```

```
public void setTipo(int t) {
tipo = t;
}
public int getTipo() {
return tipo;
}
}
```

1.7. Visibilidad

La visibilidad es la manera en que se manejan los atributos y métodos de una clase, estos pueden ser tres:

1. Público: este indica que está visible para todas las clases y puede ser utilizado por otras clases.
2. Protegido: que solo puede ser usado por la clase que lo contenga o una subclase de esta.
3. Privado: este es el más fuerte ya que solo la clase principal puede utilizar el atributo o método.

Ejemplo:

```
public class Puerta {
protected int alto;
private int color;
public void Abrir();
}
```

1.8. Estructura de datos

s una forma particular de organizar datos para que puedan ser utilizados por la computadora de manera más eficiente, consiste en una colección de datos que se caracteriza por su organización y las operaciones que se definen en ella.
Ejemplo:
Una cadena: “holi”

Capítulo 2

Errores

2.1. Errores de Sintaxis

Cuando no se puede ejecutar el programa porque alguna instrucción está mal redactada.
Ejemplos (en Python):

1. `prrint \holi"`

   No va a imprimir nada pues el print tiene una r de más.

2. ```
   r == 1
   print r
   ```

   No va a regresar nada pues se hizo una comparación en vez de una asignación

3. ```
    s = 5 ++ 7
   print s
   ```

   Va a marcar un error pues hay un + de más.

4. ```
   def saludo (x)
   if x == True :
   return \holi"
   ```

   No va a regresar nada pues después de definir la función no están los dos puntitos.

2.2. Errores de Ejecución

Se presenta cuando el ordenador no puede ejecutar alguna instrucción de forma correcta.
Ejemplos:
   ```

1. ```
   r = 1/(1/2)
   print r
   ```

   Lo que se espera es que devuelva 0.5, pero solo regresa 0 pues como los numeritos son de tipo entero lo que devuelven también será de tipo entero.

2. ```
   Edad = float(input(\Ingresa tu edad: \))
   ```

   Aquí el usuario ingresa su edad y el programa lo convierte en un flotante, todo esto suponiendo que ingresa un número, pues si ingresa por ejemplo una letra el programa marcará un error de ejecución.

2.3. Errores Lógicos

Estos son los más difíciles de detectar pues se presentan cuando no existe ningún error de sintaxis o de ejecución, pero aun así el programa no puede ejecutar el código correctamente. De manera que, un error de lógica se produce cuando los resultados obtenidos no son los esperados.
Ejemplos:

1. ```
   def AreaCuadrado (lados) :
   a = lado*lado*lado
   print a
   ```

   El programa se ejecutará sin problema, pero si por ejemplo introducimos 10 como la longitud de los lados del cuadrado nos devolverá 1000 como área, pues definimos a = lado*lado*lado, cuando debió haber sido a = lado*lado

2. ```
   def Impar (numerillo) :
   if numerillo%2 == 1 :
   return False
   else :
   return True
   ```

   Aquí no habrá ningún problema al ejecutar, sin embargo pones un 10 como entrada te dirá que sí es impar cuando no es así, esto se debe a que la comparación debe ir de la siguiente manera: if numerillo %2 == 0.

3. ```
   def AreaTriangulo (base, altura) :
   a = (base*altura)/3
   print a
   ```

   Al igual que con los ejemplos anteriores no habrá ningún problema al ejecutar pero no devolverá lo que se espera, pues a debió definirse como a = (base*altura)/2
   ```

Capítulo 3

Excepciones

3.1. ¿Qué es una excepción?

Es un mecanismo de control de errores en tiempo de ejecución, una forma de hacer que la ejecución del programa continue cuando se produce un error.

3.2. Jerarquía

Se define una excepción como un objeto que es una instancia de la clase Throwable, o alguna de sus subclases. De Throwable heredan las clases Error y Exception:

1. Exception: se usa para las excepciones que deberán capturar los programas de usuario.
2. II) Error: Excepciones que no se suelen capturar en condiciones normales. Suelen ser fallos catastróficos no gestionados por los programas, como el desbordamiento de una pila.

3.3. Excepciones definidas por el programador

1. RuntimeException: representa excepciones definidas automáticamente por los programas (división por 0, índice inválido de matriz, etc).
2. ClassNotFoundException
3. InterruptedException
4. CloneNotSuppoertedException
5. IllegalAccesException
6. NoSuchMethodException

Capítulo 4

Objeto persistente

La persistencia es la acción de mantener la información del objeto de una forma permanente (guardarla), pero también debe de poder recuperarse dicha información para que pueda ser utilizada nuevamente.
Para la persistencia los objetos podrían clasificarse en dos tipos:

1. Objetos transitorios: son aquellos que su tiempo de vida depende del espacio del proceso que lo creo.

2. Persistentes: son aquellos que su estado es almacenado en un medio temporal para su posterior reconstrucción y utilización, por lo cual el objeto no depende del proceso que lo creo.

Capítulo 5

Entornos para ejecutar código

5.1. Netbeans

Es un programa que sirve como IDE (un entorno de desarrollo integrado) que nos permite programar en diversos lenguajes.
Ventajas:

1. Lenguaje multiplataforma
2. Manejo automático de la memoria
3. Desarrolla aplicaciones web dinámicas
4. Es gratuito

Desventajas:

1. Lentitud al ejecutar aplicaciones
2. Requiere un intérprete
3. Algunas implementaciones y librerías pueden tener código rebuscado.
4. Algunas herramientas tienen un costo adicional.

5.2. Jupyter

Jupyter Notebook es un entorno de trabajo interactivo que permite desarrollar código en Python, por defecto, de manera dinámica, a la vez que integrar en un mismo documento tanto bloques de código como texto, gráficas o imágenes.
Ventajas:

1. Permite replicar resultados de una forma sencilla
2. Puedes exportar los resultados a HTML y PDF de forma inmediata.
3. Las variables quedan almacenadas en memoria para poder inspeccionar y corregir el código fácilmente.

Desventajas:

1. No es compatible con Git
2. No sirve en el desarrollo de librerías o de software serio.
3. Los Notebook pueden llegar a ser muy largos.

5.3. Emacs

Emacs es un editor de texto con una gran cantidad de funciones, es uno de los editores de texto más potentes de Unix-Linux.
Ventajas:

1. Su velocidad
2. La gran cantidad de paquetes que contiene
3. Funciona tanto en modo texto como en modo gráfico
4. Es gratuito
5. Instalación fácil

Desventajas:

1. Son muchas combinaciones de teclas
2. Curva de aprendizaje amplia