Intro POO (IV)

- Constructores y Destructores
- Ocultamiento de Datos
- Operador this

Atención!!! - recordar que...

- Si bien No se ha mencionado, a partir de lo hecho hasta aquí es posible observar que:
  - El acceso por omisión dado a un elemento miembro de una clase es **PÚBLIC**O!!!

Constructores y Destructores

- Para declarar un objeto se usa el operador **new**
- La sola declaración no sirve, el objeto debe crearse:
  - **<clase> <objeto> = new <clase>( ) ;**

Constructores y Destructores

- A diferencia de otros LDP como C++, en Java no es necesario destruir explícitamente un objeto.
- El objeto se destruye cuando *(concepto de Garbage Collector)*:
  - Se abandona el segmento de código donde se creó.
  - O cuando a su referencia se le asigna NULL

```
<objeto> = NULL;
```

Constructores y Destructores

```
class cosa {
 private int num;
 protected char letra;
 public cosa( ) {
  num = 1;
  letra = 'a';
 }
 public cosa(int n, char l ) {
  num = n;
  letra = l;
 }
 // aquí más operaciones...
```

```
// dentro del main de un programa
...main( ) {
  cosa artefacto = new cosa();
  cosa otraCosa = mew cosa();
  …// mucho código
  OtraCosa = NULL;
}
```

¿Qué pasa con el objeto artefacto?

¿Qué se hace al final con el objeto otraCosa ?

Operador This

- Se usa al igual que en C++ o MS Visual Basic, para hacer que el objeto haga referencias a sí mismo.
- En un constructor permite hacer uso de otro constructor.

```
class arreglo {
    Int [ ] vector = new int[50];

    public arreglo(int n) {
        for (int i=0;i<n;i++)
            this.vector[i] = i;
    }
…..
}
```

¿Qué sucede aquí?

Ejemplo de código

```
// Clase que trabaja con el uso de operador this
// implementa el uso de arreglos
// hace uso de constructores y la destruccion explicita de objetos con NULL

import java.io.*;

public class Test {
	public static void main(String [] args) throws Exception {
		InputStreamReader isr = new InputStreamReader(System.in);
		BufferedReader br = new BufferedReader(isr);

		System.out.println("\nTrabajo basico con Arreglos!!!\n");
		System.out.println("Cuantos datos quieres mostrar? : ");

		int largo = Integer.parseInt(br.readLine());

		arreglo A = new arreglo(largo);
		System.out.println("Largo total arreglo: "+A.vector.length);
		A.mostrar(largo);

		// destruccion explicta del objeto
		A = null;
		if (A==null) System.out.println("Objeto destruido\n");
	}
}
```

Ejemplo de Código

```
class arreglo {
	int [] vector = new int[50];

	public arreglo(int n) {
		for (int i=0;i<n;i++)
			this.vector[i] = i;
	}

	public void mostrar(int n) {
		for (int i=0;i<n;i++)
		System.out.println(vector[i]);
	}
}
```

- ¿Qué sucede al ejecutar la instrucción **A = null;**?

- ¿Qué sucede si ejecuto ***A.mostrar(largo)*** luego de invocar al destructor del objeto A?

Descargar el ejemplo Test.java en Campus Virtual

Ocultamiento de Información

- Cuando se crea una nueva clase en Java, se puede especificar el nivel de acceso que se quiere para:
  - las variables de instancia y
  - los métodos definidos en la clase
- Lo anterior, se puede hacer mediante los denominados modificadores de acceso.
- En el caso de Java y otro LDP estos se definen a través de los operadores

→ **public, protected, private.**

Ocultamiento de información

- **Public**: Cualquier clase puede acceder a las propiedades y métodos públicos.
- **Protected**: Sólo las clases heredades y aquellas situadas en el mismo paquete pueden acceder a las propiedades y métodos protegidos.
- **Private**: Las variables y métodos privados sólo pueden ser accedidos desde dentro de la clase.

Ej. Ocultamiento - usando la clase Date

- Permite trabajar con datos del tipo fecha.
- Para crerar objetos del tipo Date, se debe recurrir a realizar un **import**
  import java.util.*
- Date fecha = new Date();
- Incorpora un set de funciones para “recuperar” y “asignar” valores componentes de una fecha:
  - getYear(), getMinute, etc...
  - setYear(), setMinute(), etc...

Ej. Ocultamiento - usando la clase Date

```
public class TodaysDate {
  //member variables
  String time;
  public int day;
  private int month;
  protected int year;

  //methods
  public void printDateAndTime(){
    GregorianCalendar calendar = new GregorianCalendar();
    time = calendar.get(Calendar.HOUR_OF_DAY) + ":"
    + calendar.get(Calendar.MINUTE) + ":"
    + calendar.get(Calendar.SECOND);
    day = calendar.get(Calendar.DATE);
    month = calendar.get(Calendar.MONTH) + 1;
    year = calendar.get(Calendar.YEAR);

    out.println("Time: " + time);
    out.println("Date: "+ month + " " + day + " " + year);
  }
```

Ej. Ocultamiento - usando la clase Date

```
public class TodaysDate {
  //member variables
  String time;
  public int day;
  private int month;
  protected int year;

  //methods
  public void printDateAndTime(){
    GregorianCalendar calendar = new GregorianCalendar();
    time = calendar.get(Calendar.HOUR_OF_DAY) + ":"
    + calendar.get(Calendar.MINUTE) + ":"
    + calendar.get(Calendar.SECOND);
    day = calendar.get(Calendar.DATE);
    month = calendar.get(Calendar.MONTH) + 1;
    year = calendar.get(Calendar.YEAR);

    out.println("Time: " + time);
    out.println("Date: "+ month + " " + day + " " + year);
  }
```

Ej. Ocultamiento - usando la clase Date

- Trabajo en clases
  - Ud. y otro compañero, dado el código anterior, desarrolle la siguiente actividad.
  - Responda las siguientes preguntas, seleccionando la(s) opción(es) correctas en cada caso, justificando sus respuestas.
  - Tiempo: 30 minutos, entregue sus resultados en forma escrita (puede ser una hoja de cuaderno, ordenada y bien redactada).

Ej. Ocultamiento - usando la clase Date

- Cree una clase pública llamada TestFecha
- Dicha clase en su método main() cree:
  - Un objeto llamado ***hoy*** del tipo TodaysDate
  - Use el método *printDateAndTime()* de su objeto *hoy*
- Su salida por pantalla debiera ser como lo que se muestra, tras compilar y ejecutar su código.

```
Time: 15:43:13
Date: 5 11 2014
```

Ej. Ocultamiento - usando la clase Date

- **Pgta 1.** ¿Desde dónde es posible acceder a la variable ***time***?

1. Cualquier otra clase
2. java.util package
3. myUtilities package
4. La clase TodaysDate
5. La clase TodaysDate y sus sub clases

Ej. Ocultamiento - usando la clase Date

- **Pgta 2.**

  ¿Quiénes pueden acceder al atributo ***day***?

Ej. Ocultamiento - usando la clase Date

- **Pgta 3.** ¿Qué atributos de la clases tienen el método de acceso más restrictivo?

1. day
2. month
3. time
4. year

Ej. Ocultamiento - usando la clase Date

- **Pgta 4.** ¿Desde dónde se puede acceder al atributo ***year***?

1. Sólo dentro del package myUtilities.
2. Sub clases de TodaysDate en cualquier package
3. La clase TodaysDate
4. Sub clases de TodaysDate

Resumiendo

- Constructores
- Destructores
- Operador This
- Ocultamiento de información

Próximos temas

- Herencia
- Herencia y ocultamiento de datos
- Herencia y operador Super
- Mensajes y asociaciones entre clases