Sprint 4 Summary

Doscker

15 de octubre de 2018

Actualización de Journals

Sprint 3 Summary

Doscker

30 de septiembre de 2018

Estructuras de control básicas

Ejercicio

Diseñe un programa que solicite un número al usuario e indique si dicho numero es perfecto o no.

```
import java.util.Scanner;
public class nperfectos {

  public static void main(String[] args) {

    //Verificar si el numero ingresado es perfecto

    Scanner input=new Scanner(System.in);
    System.out.println("Ingresa un numero");

    int a=Integer.parseInt(input.next());
    int cont=0;

    for(int i=1; i<a; i++){
      if(a%i==0){
        cont+=i;
      }
    }

    if(cont==a){
      System.out.println("Es perfecto");
    }else{
      System.out.println("No es perfecto");
    }
  }
}
```

```
import java.util.Scanner;
public class nperfectos2 {

  public static void main(String[] args) {
```

Figura 1: Para ver los Jornals pasados completos, visitar: https://github.com/Fernando0107/Doscker

Ampliar los conocimientos obtenidos V1

```java
package com.jetbrains;
import java.util.Scanner;
import java.io.*;
import java.util.Arrays;

public class Files {

    public static void main(String[]args) throws IOException {
        try {
            Scanner in = new Scanner(new FileReader( fileName: "notas.txt"));          // Lectura del archivo
            int nota1=Integer.parseInt(in.nextLine());
            int nota2=Integer.parseInt(in.nextLine());
            int nota3=Integer.parseInt(in.nextLine());
            int nota4=Integer.parseInt(in.nextLine());
            int nota5=Integer.parseInt(in.nextLine());
            int nota6=Integer.parseInt(in.nextLine());
            int nota7=Integer.parseInt(in.nextLine());
            int nota8=Integer.parseInt(in.nextLine());
            int nota9=Integer.parseInt(in.nextLine());
            int nota10=Integer.parseInt(in.nextLine());
            int nota11=Integer.parseInt(in.nextLine());
            int promedio=((nota1+nota2+nota3+nota4+nota5+nota6+nota7+nota8+nota9+nota10)/10);
            System.out.println("Promedio = "+ promedio);

            int num[]={nota1,nota2,nota3,nota4, nota5,nota6,nota7, nota8,nota9,nota10,nota11}; // Guardo las notas en un array
```

```
Files › main()
Files ×
"C:\Program Files\Java\jdk-10.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2018.2\
Promedio = 100
Maximo = 100
Minimo = 100
La desviación estandar es : 0.0

Process finished with exit code 0
```

Figura 2: Para ver y ejecutar el programa, visitar: https://github.com/Fernando0107/Doscker/blob/master/R

Puedo explicar la diferencia entre clase y objeto

Destructor y constructor de clase

```java
import java.util.Date; // (https://docs.oracle.com/javase/7/docs/api/)
import java.text.SimpleDateFormat; // (https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html)

public class Task{

    /* Do not modify */
    private String msg_reminder, notes;
    private Date do_date;
    private int priority = 0;
    private boolean status = false;
    private String pattern = "yyyy-MM-dd";
    private SimpleDateFormat simpleDateFormat;

    /* Task con msg_reminder, prioridad y date*/
    public Task(String msg_reminder, Date do_date, int priority) {
        this.msg_reminder = msg_reminder;
        this.do_date = do_date;
        this.priority = priority;
    }
    /* Task con fecha, prioridad y notas */
    public Task(String msg_reminder, Date do_date, int priority, String notes) {
        this.msg_reminder = msg_reminder;
        this.do_date = do_date;
        this.priority = priority;
        this.notes = notes;

    }
    /* Task con msg_reminder y date */
    public Task(String msg_reminder) {
        this.msg_reminder = msg_reminder;
        this.do_date = getTaskDate();
```

Figura 3: Para ver y ejecutar el programa, visitar: https://github.com/Fernando0107/Doscker/blob/master/T

Conceptos de Overloading

```java
import java.lang.Math;
import java.util.Scanner;

public class Cajero
{
    public int total(int x, int y) {
        return (x + y);
    }
    public int total(int x, int y, int z) {
        return (x + y + z);
    }
    public int total(int x, int y, int z, int w) {
        return (x + y + z + w);
    }

    public static void main(String[] args)
    {
        int a;
        int b;
        int c;
        int d;
        Scanner user = new Scanner(System.in);
        Cajero t = new Cajero();
        System.out.println("Ingrese el precio de los productos (max 4) (ingresar 0 si termino)");
        a = user.nextInt();
        b = user.nextInt();
        if (b == 0) {
            System.out.println("Su total es: Q." + a);
            System.exit(0);
        }
```

Figura 4: Para ver y ejecutar el programa, visitar: https://github.com/Fernando0107/Doscker/blob/master/C

Asignación y clonación de objetos

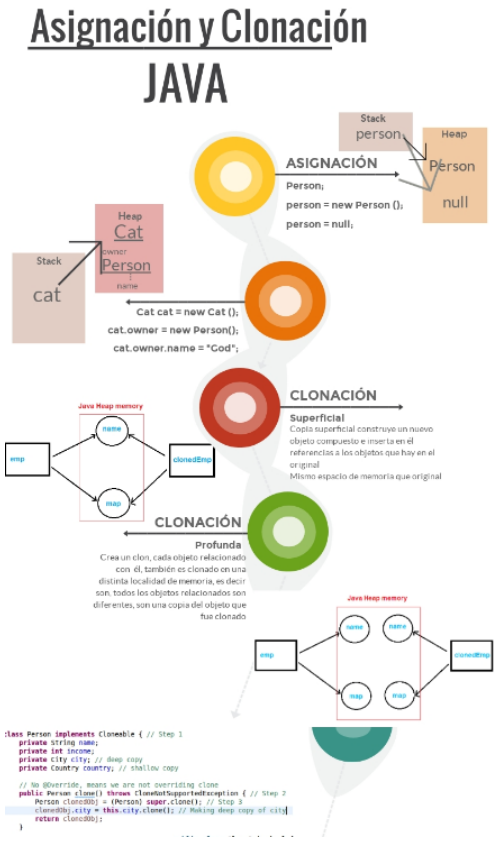


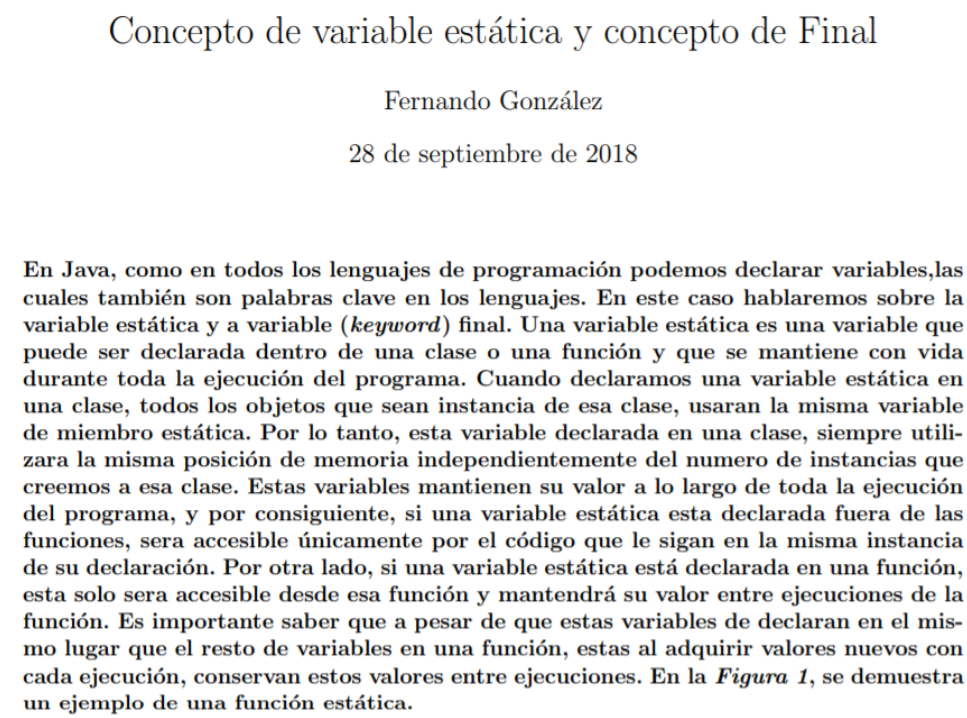
Figura 5: Para ver la imagen completa, visitar: https://github.com/Fernando0107/Doscker/blob/master/Asig y-clonacin-JAVA.jpg

Recursión en PO

```java
import java.util.Scanner;

public class Recursion_1 {

    private static Integer recursion(int n) {
        if (n == 0) return 0;
        return n + recursion(n - 1);
    }
    public static void main(String[] args) {
        int n;
        Scanner user = new Scanner(System.in);
        System.out.println("Ingrese un número: ");
        n = user.nextInt();
        System.out.println(recursion(n));

    }
}
```

Figura 6: Para ver los códigos completos y ejecutarlos, visitar: https://github.com/Fernando0107/Doscker/blob/master/Recursion.zip

Ensayo sobre variable estática y final



Concepto de variable estática y concepto de Final

Fernando González

28 de septiembre de 2018

En Java, como en todos los lenguajes de programación podemos declarar variables,las cuales también son palabras clave en los lenguajes. En este caso hablaremos sobre la variable estática y a variable (*keyword*) final. Una variable estática es una variable que puede ser declarada dentro de una clase o una función y que se mantiene con vida durante toda la ejecución del programa. Cuando declaramos una variable estática en una clase, todos los objetos que sean instancia de esa clase, usaran la misma variable de miembro estática. Por lo tanto, esta variable declarada en una clase, siempre utilizara la misma posición de memoria independientemente del numero de instancias que creemos a esa clase. Estas variables mantienen su valor a lo largo de toda la ejecución del programa, y por consiguiente, si una variable estática esta declarada fuera de las funciones, sera accesible únicamente por el código que le sigan en la misma instancia de su declaración. Por otra lado, si una variable estática está declarada en una función, esta solo sera accesible desde esa función y mantendrá su valor entre ejecuciones de la función. Es importante saber que a pesar de que estas variables de declaran en el mismo lugar que el resto de variables en una función, estas al adquirir valores nuevos con cada ejecución, conservan estos valores entre ejecuciones. En la *Figura 1*, se demuestra un ejemplo de una función estática.

Figura 7: Para ver el ensayo presentado en el documento, visitar: https://github.com/Fernando0107/Doscker/blob/master/Ensayo.pdf (El resto de ensayos fueron entregados por medio de MiU)