

NOTAS DE IPO

Plantilla de notas

ENRIQUE M. DELGADO TORRES

2023

ÍNDICE GENERAL

1

Capítulo 1

Introducción a la Interacción Persona Ordenador

- 1.1 Concepto Interfaz en el contexto de la IPO 3
- 1.2 ¿Qué es la IPO? 3
- 1.3 Disciplinas Integradas en la IPO 4
 - 1.3.1 Diseño 4
 - 1.3.2 Psicología 4
 - 1.3.3 Ergonomía 4
 - 1.3.4 Programación 4

INTRODUCCIÓN A LA INTERACCIÓN PERSONA ORDENADOR

1

1.1

Concepto Interfaz en el contexto de la IPO

La interfaz refleja las **Propiedades Físicas**, las **Funciones** y el **Balance de Poder y Control**, facilitando así la transmisión de información, órdenes y datos. Además, permite compartir sensaciones, intuiciones y nuevas formas de ver las cosas entre humanos y sistemas informáticos.

- **Propiedades Físicas:** La interfaz está diseñada teniendo en cuenta tanto las necesidades físicas y ergonómicas del usuario como las características del sistema informático, garantizando comodidad y accesibilidad.
- **Funciones:** La interfaz muestra y facilita las tareas específicas que el usuario necesita realizar, asegurando una interacción intuitiva y eficiente.
- **Balance de Poder y Control:** La interfaz define quién controla la interacción, ya sea el usuario o el sistema, influyendo en la experiencia del usuario y la eficacia del sistema.

1.2

¿Qué es la IPO?

La Interacción Persona-Ordenador (IPO) es una disciplina multidisciplinaria dedicada al diseño, evaluación e implementación de interfaces que faciliten una interacción entre seres humanos y sistemas informáticos de forma segura, efectiva, útil, eficiente, usable, accesible e inclusiva.

La IPO se enfoca en la adaptabilidad, asegurando que las interfaces se desarrollen y evolucionen según las necesidades cambiantes de los usuarios y los avances tecnológicos.

Traslada los conocimientos de diversas disciplinas para desarrollar herramientas y técnicas que ayuden a los diseñadores a crear sistemas informáticos idóneos.

1.3

Disciplinas Integradas en la IPO

La IPO combina conocimientos y métodos de diversas disciplinas, incluyendo:

- Diseño
- Programación
- Sociología
- Psicología
- Ingeniería de Software
- Ergonomía
- Inteligencia Artificial

1.3.1. Diseño

Disciplina centrada en crear interfaces que son tanto funcionales como estéticamente agradables, buscando mejorar la experiencia y el entorno del usuario mediante soluciones intuitivas, accesibles y atractivas.

1.3.2. Psicología

Disciplina centrada en estudiar cómo los individuos y grupos procesan información, se comportan y toman decisiones al interactuar con sistemas informáticos.

Cabe distinguir entre:

- **La Psicología Cognitiva** se enfoca en entender procesos mentales individuales, como percepción y toma de decisiones.
- **La Psicología Social** examina cómo el entorno social afecta el comportamiento del usuario. Ambas ramas contribuyen a diseñar interfaces más intuitivas, eficientes y satisfactorias para los usuarios.

1.3.3. Ergonomía

Disciplina centrada en el diseño de interfaces y entornos de trabajo que maximizan la comodidad, eficiencia y seguridad. Esto incluye la organización óptima de controles y pantallas, consideración de factores físicos como la iluminación y la posición, y el uso adecuado de colores para facilitar la interacción y prevenir riesgos de salud.

1.3.4. Programación

La programación es el proceso de diseñar y codificar programas de computadora para resolver problemas o realizar tareas específicas. Se fundamenta en diversos paradigmas, cada uno con su enfoque y metodología distintiva:

- **Programación Orientada a Objetos:** Emplea clases y objetos para estructurar el código. Facilita la modularidad y reutilización.

Ejemplo: Java, Python.

Listing 1.1: Ejemplo de POO en Java

```
1  // Definicion de la clase "Coche"
2  // Una clase es una plantilla para crear objetos
3  public class Coche {
4      // Atributos de la clase
5      // Representan las características del objeto
6      private String marca;
7      private int ano;
8
9      // Constructor de la clase
10     // Inicializa un nuevo objeto de la clase Coche
11     public Coche(String marca, int ano) {
12         this.marca = marca;
13         this.ano = ano;
14     }
15
16     // Metodos de la clase
17     // Definen el comportamiento del objeto
18     public String getMarca() {
19         return marca;
20     }
21
22     public int getAno() {
23         return ano;
24     }
25 }
26
27 // Clase principal para ejecutar el programa
28 public class Main {
29     public static void main(String[] args) {
30         // Creacion de un objeto "Coche"
31         // Un objeto es una instancia de una clase
32         Coche miCoche = new Coche("Toyota", 2021);
33
34         // Uso de metodos del objeto
35         System.out.println("Marca del Coche: " + miCoche.getMarca());
36         System.out.println("Ano del Coche: " + miCoche.getAno());
37     }
38 }
```

- **Programación Imperativa:** Centrada en la secuencia de comandos para manipular el estado de las variables.

Ejemplo: C, Pascal.

Listing 1.2: Ejemplo de Programacion Imperativa en C

```
1 #include <stdio.h>
2
3 int main() {
4     int contador = 0; // Inicializa una variable
5
6     // Ciclo para incrementar la variable
7     for (int i = 0; i < 5; i++) {
8         contador += i;
9         printf("Valor actual del contador: %d\n", contador);
10    }
11
12    return 0;
13 }
```

Salida esperada:

Valor actual del contador: 0
Valor actual del contador: 1
Valor actual del contador: 3
Valor actual del contador: 6
Valor actual del contador: 10

- **Programación Funcional:** Aborda los problemas mediante funciones, enfocándose en las relaciones de entrada y salida.

Ejemplo: Haskell, Lisp.

Listing 1.3: Ejemplo de Programacion Funcional en Haskell

```
1 -- Define una funcion simple que duplica un numero
2 duplicar :: Int -> Int
3 duplicar x = x * 2
4
5 -- Funcion principal
6 main :: IO ()
7 main = print (duplicar 5)
```

Salida esperada:

10

- **Programación Declarativa:** Se concentra en describir el *qué* de las operaciones, dejando el *cómo* a la interpretación del sistema.

Ejemplo: SQL, Prolog.

Listing 1.4: Ejemplo de Programacion Declarativa en SQL

```
1 -- SQL para seleccionar nombres de una tabla de empleados
2 SELECT nombre FROM empleados WHERE edad > 30;
```

Salida esperada:

[Lista de nombres de empleados mayores de 30 años]

- **Programación Concurrente:** Maneja operaciones que se ejecutan simultáneamente, esencial en aplicaciones multitarea y multiusuario.

Ejemplo: Erlang, Go.

Listing 1.5: Ejemplo de Programacion Concurrente en Go

```
1 package main
2
3 import (
4     "fmt"
5     "time"
6 )
7
8 // Funcion que se ejecutara de manera concurrente
9 func imprimirNumeros() {
10     for i := 1; i <= 5; i++ {
11         time.Sleep(1 * time.Second)
12         fmt.Println(i)
13     }
14 }
15
16 func main() {
17     // Ejecutar la funcion imprimirNumeros de manera concurrente
18     go imprimirNumeros()
19
20     // Esperar a que el usuario presione una tecla
21     fmt.Println("Presiona Enter para finalizar")
22     fmt.Scanln()
23 }
```

Salida esperada:

Presiona Enter para finalizar
[Los números del 1 al 5 se imprimirán uno cada segundo]