NOTAS DE IPO

Plantilla de notas

Enrique M. Delgado Torres

2023

ÍNDICE GENERAL

CAPÍTULO

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

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

```
// Definicion de la clase "Coche"
// Una clase es una plantilla para crear objetos
public class Coche {
    // Atributos de la clase
    // Representan las caracteristicas del objeto
    private String marca;
    private int ano;

    // Constructor de la clase
    // Inicializa un nuevo objeto de la clase Coche
    public Coche(String marca, int ano) {
        this.marca = marca;
        this.ano = ano;
    }

    // Metodos de la clase
    // Definen el comportamiento del objeto
    public String getMarca() {
        return marca;
    }

    public int getAno() {
        return ano;
    }
}

// Clase principal para ejecutar el programa
public class Main {
    public static void main(String[] args) {
        // Creacion de un objeto "Coche"
        // Un objeto es una instancia de una clase
        Coche miCoche = new Coche("Toyota", 2021);

        // Uso de metodos del objeto
        System.out.println("Marca del Coche: " + miCoche.getMarca
            ());
        System.out.println("Ano del Coche: " + miCoche.getAno());
    }
}
```

Explicación del Código: El código anterior demuestra un uso básico de la Programación Orientada a Objetos (POO) en Java. Se define una clase ‘Coche‘ con atributos como ‘marca‘ y ‘ano‘, y métodos para obtener estos valores. Se crea un objeto ‘miCoche‘ de esta clase y se utilizan sus métodos para imprimir la marca y el año. Este enfoque modular y reutilizable de la POO facilita la organización y escalabilidad del código, permitiendo la creación de múltiples instancias de ‘Coche‘ con diferentes características.

Output del Programa:

```
Marca del Coche: Toyota
Ano del Coche: 2021
```

- **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

```c
#include <stdio.h>

int main() {
    // Declaracion de variables
    int numero = 10; // Una variable 'numero' inicializada en 10
    int factor = 2;  // Una variable 'factor' inicializada en 2
    int resultado;   // Una variable 'resultado' sin inicializar

    // Mostrar el valor inicial de 'numero'
    printf("Valor inicial de numero: %d\n", numero);

    // Secuencia de comandos para manipular las variables
    resultado = numero * factor; // Multiplicar 'numero' por 'factor'

    // Mostrar el resultado de la operacion
    printf("Resultado de %d multiplicado por %d es: %d\n", numero, factor, resultado);

    return 0;
}
```

Explicación del Código: Este código demuestra la programación imperativa en C. Comienza con la declaración e inicialización de dos variables, 'numero' y 'factor'. Luego, una tercera variable 'resultado' es declarada. La secuencia de comandos incluye una operación de multiplicación, donde 'numero' es multiplicado por 'factor', y el resultado se almacena en 'resultado'. Finalmente, el programa imprime tanto el valor inicial de 'numero' como el 'resultado' de la operación.
Salida esperada:

```
Valor inicial de numero: 10
Resultado de 10 multiplicado por 2 es: 20
```

- **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

```haskell
-- Define una funcion simple que duplica un numero
duplicar :: Int -> Int
duplicar x = x * 2

-- Funcion principal
main :: IO ()
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

```sql
-- SQL para seleccionar nombres de una tabla de empleados
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

```go
package main

import (
    "fmt"
    "time"
)

// Funcion que se ejecutara de manera concurrente
func imprimirNumeros() {
    for i := 1; i <= 5; i++ {
        time.Sleep(1 * time.Second)
        fmt.Println(i)
    }
}

func main() {
    // Ejecutar la funcion imprimirNumeros de manera concurrente
    go imprimirNumeros()

    // Esperar a que el usuario presione una tecla
    fmt.Println("Presiona Enter para finalizar")
    fmt.Scanln()
}
```

Salida esperada:

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