Practica 4. Herencia

Hernández Rentería Jorge Andrés

2025-09-22

1. Crear una clase Ticket con los siguientes atributos: -id -tipo (por ejemplo: software, prueba) -prioridad (alta, media, baja) -estado (por defecto ‘pendiente’)
2. Crear dos tickets de ejemplo y mostrarlos por pantalla.
3. Crear padre (clase) Empleado
   a) Crear la clase Empleado con atributo nombre.
   b) Crear método trabajar_en_ticket(self,ticket) que imprima: “El empleado revisa el ticket “.
4. Crear clase Desarrollador que herede de Empleado y sobrescriba el método trabajar_en_ticket:

- Solo puede resolver tickets de tipo “Software”. (validación)
- Si lo hace, cambia el estado a “resuelto” y muestra un mensaje.

5. Crear clase Tester que solo pueda resolver tickets de tipo “prueba”. (condición)
6. Crear clase ProjectManager que asigne tickets

a) Crear la clase ProjectManager que herede de Empleado.
b)

#Parte adicional:

Agregar un menú con while y con un if que permita: 1.- Crear un ticket 2.- Ver tickets 3.- Asignar tickets 4.- Salir del programa

```
"""
class Ticket:
    def __init__(self, id, tipo, prioridad):
        self.id = id
        self.tipo = tipo
        self.prioridad = prioridad
        self.estado = "pendiente"

    def __str__(self):
        return f"Ticket {self.id}: {self.tipo} - Prioridad: {self.prioridad} -
Estado: {self.estado}"
```

```
class Empleado:
    def __init__(self, nombre):
        self.nombre = nombre

    def trabajar_en_ticket(self, ticket):
        print(f"El empleado {self.nombre} revisa el ticket {ticket.id}")

class Desarrollador(Empleado):
    def trabajar_en_ticket(self, ticket):
        if ticket.tipo.lower() == "software":
            ticket.estado = "resuelto"
            print(f"El ticket {ticket.id} fue resuelto por {self.nombre}")
        else:
            print("Este tipo de empleado no puede resolver este ticket")

class Tester(Empleado):
    def trabajar_en_ticket(self, ticket):
        if ticket.tipo.lower() == "prueba":
            ticket.estado = "resuelto"
            print(f"El ticket {ticket.id} fue resuelto por {self.nombre}")
        else:
            print("Este tipo de empleado no puede resolver este ticket")

class Project_manager(Empleado):
    def asignar_ticket(self, ticket, empleado):
        print(f"{self.nombre} asignó el ticket {ticket.id} al empleado:
{empleado.nombre}")
        empleado.trabajar_en_ticket(ticket)

# Sistema de gestión
class SistemaTickets:
    def __init__(self):
        self.tickets = []
        self.empleados = []
        self.contador_tickets = 1
        self.cargar_datos_ejemplo()

    def cargar_datos_ejemplo(self):
        # Crear empleados de ejemplo
        self.empleados.append(Desarrollador("Jorge"))
        self.empleados.append(Tester("Pablo"))
        self.empleados.append(Project_manager("Susana"))

        # Crear tickets de ejemplo
        self.tickets.append(Ticket(1, "software", "alta"))
        self.tickets.append(Ticket(2, "prueba", "media"))
        self.contador_tickets = 3
```

```
    def crear_ticket(self):
        print("\n--- CREAR NUEVO TICKET ---")

        # Validar tipo
        while True:
            tipo = input("Tipo (software/prueba): ").lower()
            if tipo in ["software", "prueba"]:
                break
            else:
                print("Error: Tipo debe ser 'software' o 'prueba'")

        # Validar prioridad
        while True:
            prioridad = input("Prioridad (alta/media/baja): ").lower()
            if prioridad in ["alta", "media", "baja"]:
                break
            else:
                print("Error: Prioridad debe ser 'alta', 'media' o 'baja'")

        # Crear ticket
        nuevo_ticket = Ticket(self.contador_tickets, tipo, prioridad)
        self.tickets.append(nuevo_ticket)
        self.contador_tickets += 1

        print(f"Ticket {nuevo_ticket.id} creado exitosamente!")
        return nuevo_ticket

    def ver_tickets(self):
        print("\n--- LISTA DE TICKETS ---")
        if not self.tickets:
            print("No hay tickets creados.")
            return

        for ticket in self.tickets:
            print(ticket)

    def asignar_ticket(self):
        print("\n--- ASIGNAR TICKET ---")

        # Verificar que hay tickets
        if not self.tickets:
            print("No hay tickets para asignar.")
            return

        # Mostrar tickets disponibles
        print("Tickets disponibles:")
        for i, ticket in enumerate(self.tickets, 1):
            print(f"{i}. {ticket}")
```

```
        # Seleccionar ticket
        try:
            seleccion = int(input("Selecciona el número del ticket: ")) - 1
            if 0 <= seleccion < len(self.tickets):
                ticket_seleccionado = self.tickets[seleccion]
            else:
                print("Selección inválida.")
                return
        except ValueError:
            print("Ingresa un número válido.")
            return

        # Mostrar empleados disponibles
        print("\nEmpleados disponibles:")
        for i, empleado in enumerate(self.empleados, 1):
            tipo_empleado = type(empleado).__name__
            print(f"{i}. {empleado.nombre} ({tipo_empleado})")

        # Seleccionar empleado
        try:
            seleccion_emp = int(input("Selecciona el número del empleado: "))
- 1
            if 0 <= seleccion_emp < len(self.empleados):
                empleado_seleccionado = self.empleados[seleccion_emp]
            else:
                print("Selección inválida.")
                return
        except ValueError:
            print("Ingresa un número válido.")
            return

        # Buscar Project Manager para asignar
        pm = None
        for empleado in self.empleados:
            if isinstance(empleado, Project_manager):
                pm = empleado
                break

        if pm:
            pm.asignar_ticket(ticket_seleccionado, empleado_seleccionado)
        else:
            print("No hay Project Manager disponible.")

    def menu_principal(self):
        while True:
            print("\n" + "="*50)
            print("        SISTEMA DE GESTIÓN DE TICKETS")
```

```
            print("="*50)
            print("1. Crear un ticket")
            print("2. Ver tickets")
            print("3. Asignar tickets")
            print("4. Salir del programa")
            print("-"*50)

            opcion = input("Selecciona una opción (1-4): ")

            if opcion == "1":
                self.crear_ticket()
            elif opcion == "2":
                self.ver_tickets()
            elif opcion == "3":
                self.asignar_ticket()
            elif opcion == "4":
                print("¡Gracias por usar el sistema!")
                break
            else:
                print("Opción inválida. Por favor, selecciona 1-4.")

            # Pausa antes de continuar
            input("\nPresiona Enter para continuar...")

# Ejecutar el sistema
if __name__ == "__main__":
    sistema = SistemaTickets()
    sistema.menu_principal()

"""
```

```
'\nclass Ticket:\n    def __init__(self, id, tipo, prioridad):\n
self.id = id \n        self.tipo = tipo\n        self.prioridad = prioridad\n
self.estado = "pendiente"\n\n    def __str__(self):\n        return f"Ticket
{self.id}: {self.tipo} - Prioridad: {self.prioridad} - Estado:
{self.estado}"\n\nclass Empleado:\n    def __init__(self, nombre):\n
self.nombre = nombre\n\n    def trabajar_en_ticket(self, ticket):\n
print(f"El empleado {self.nombre} revisa el ticket {ticket.id}")\n\nclass
Desarrollador(Empleado):\n    def trabajar_en_ticket(self, ticket):\n
if ticket.tipo.lower() == "software":\n            ticket.estado = "resuelto"
\n            print(f"El ticket {ticket.id} fue resuelto por {self.nombre}")\n
else:\n            print("Este tipo de empleado no puede resolver este
ticket")\n\nclass Tester(Empleado):\n    def trabajar_en_ticket(self, ticket):
\n        if ticket.tipo.lower() == "prueba":\n            ticket.estado =
"resuelto"  \n            print(f"El ticket {ticket.id} fue resuelto por
{self.nombre}")\n        else:\n            print("Este tipo de empleado no
puede resolver este ticket")\n\nclass Project_manager(Empleado):\n    def
```

```
asignar_ticket(self, ticket, empleado):\n        print(f"{self.nombre} asignó
el ticket {ticket.id} al empleado: {empleado.nombre}")\n
empleado.trabajar_en_ticket(ticket)\n\n# Sistema de gestión\nclass
SistemaTickets:\n    def __init__(self):\n        self.tickets = []\n
self.empleados = []\n        self.contador_tickets = 1\n
self.cargar_datos_ejemplo()\n\n    def cargar_datos_ejemplo(self):\n        #
Crear empleados de ejemplo\n
self.empleados.append(Desarrollador("Jorge"))\n
self.empleados.append(Tester("Pablo"))\n
self.empleados.append(Project_manager("Susana"))\n\n        # Crear tickets de
ejemplo\n        self.tickets.append(Ticket(1, "software", "alta"))\n
self.tickets.append(Ticket(2, "prueba", "media"))\n
self.contador_tickets = 3\n\n    def crear_ticket(self):\n        print("\n---
CREAR NUEVO TICKET ---")\n\n        # Validar tipo\n        while True:\n
tipo = input("Tipo (software/prueba): ").lower()\n            if tipo in
["software", "prueba"]:\n                break\n            else:\n
print("Error: Tipo debe ser \'software\' o \'prueba\'")\n\n        # Validar
prioridad\n        while True:\n            prioridad = input("Prioridad
(alta/media/baja): ").lower()\n            if prioridad in ["alta", "media",
"baja"]:\n                break\n            else:\n
print("Error: Prioridad debe ser \'alta\', \'media\' o \'baja\'")\n\n        #
Crear ticket\n        nuevo_ticket = Ticket(self.contador_tickets, tipo,
prioridad)\n        self.tickets.append(nuevo_ticket)\n
self.contador_tickets += 1\n\n        print(f"Ticket {nuevo_ticket.id} creado
exitosamente!")\n        return nuevo_ticket\n\n    def ver_tickets(self):\n
print("\n--- LISTA DE TICKETS ---")\n        if not self.tickets:\n
print("No hay tickets creados.")\n            return\n\n        for ticket in
self.tickets:\n            print(ticket)\n\n    def asignar_ticket(self):\n
print("\n--- ASIGNAR TICKET ---")\n\n        # Verificar que hay tickets\n
if not self.tickets:\n            print("No hay tickets para asignar.")\n
return\n\n        # Mostrar tickets disponibles\n        print("Tickets
disponibles:")\n        for i, ticket in enumerate(self.tickets, 1):\n
print(f"{i}. {ticket}")\n\n        # Seleccionar ticket\n        try:\n
seleccion = int(input("Selecciona el número del ticket: ")) - 1\n
if 0 <= seleccion < len(self.tickets):\n                ticket_seleccionado =
self.tickets[seleccion]\n            else:\n                print("Selección
inválida.")\n                return\n        except ValueError:\n
print("Ingresa un número válido.")\n            return\n\n        # Mostrar
empleados disponibles\n        print("\nEmpleados disponibles:")\n        for
i, empleado in enumerate(self.empleados, 1):\n            tipo_empleado =
type(empleado).__name__\n            print(f"{i}. {empleado.nombre}
({tipo_empleado})")\n\n        # Seleccionar empleado\n        try:\n
seleccion_emp = int(input("Selecciona el número del empleado: ")) - 1\n
if 0 <= seleccion_emp < len(self.empleados):\n
empleado_seleccionado = self.empleados[seleccion_emp]\n            else:\n
print("Selección inválida.")\n                return\n        except
ValueError:\n            print("Ingresa un número válido.")\n
return\n\n        # Buscar Project Manager para asignar\n        pm = None\n
```

```
for empleado in self.empleados:\n            if isinstance(empleado,
Project_manager):\n                pm = empleado\n                break\n\n
if pm:\n            pm.asignar_ticket(ticket_seleccionado,
empleado_seleccionado)\n        else:\n            print("No hay Project
Manager disponible.")\n\n    def menu_principal(self):\n        while True:\n
print("\n" + "="*50)\n            print("        SISTEMA DE GESTIÓN DE
TICKETS")\n            print("="*50)\n            print("1. Crear un
ticket")\n            print("2. Ver tickets")\n            print("3. Asignar
tickets")\n            print("4. Salir del programa")\n
print("-"*50)\n\n            opcion = input("Selecciona una opción (1-4):
")\n\n            if opcion == "1":\n                self.crear_ticket()\n
elif opcion == "2":\n                self.ver_tickets()\n            elif
opcion == "3":\n                self.asignar_ticket()\n            elif opcion
== "4":\n                print("¡Gracias por usar el sistema!")\n
break\n            else:\n                print("Opción inválida. Por favor,
selecciona 1-4.")\n\n            # Pausa antes de continuar\n
input("\nPresiona Enter para continuar...")\n\n# Ejecutar el sistema\nif
__name__ == "__main__":\n    sistema = SistemaTickets()\n
sistema.menu_principal()\n\n'
```