Practica 4. Herencia

Castillo Dominguez Sonia Vanessa

2026-10-09

---

title: “Practica 4. Herencia” author: “Castillo Dominguez Sonia Vanessa” date: “22/09/25” format: pdf

---

Introduccion

se implementó un sistema de simulación de manejo de tickets utilizando los conceptos de herencia y polimorfismo en el lenguaje de programación Python. El objetivo principal fue aplicar la programación orientada a objetos para modelar cómo diferentes tipos de empleados interactúan con tickets de soporte, dependiendo de su rol dentro de un equipo de desarrollo de software.

Desarrollo

ticket:Define los atributos principales de un ticket: id, tipo, prioridad y estado. clase padre: en este caso empleado: Representa a un empleado genérico. Incluye un método trabajar_ticket que puede ser sobrescrito clases hijas que son Desarrollador, ProjectManager.

Problema

crear una clase ticket con la siguientes atributos id tipo(por ejemplo: software,prueba) prioridad (alta, media , baja) Estado (por defecto “pendiente”) 2.Crear dos ticketts de ejemplo y mostrarlos por pantalla Agreagar un menu interactivo con while y con if para: Crear un ticket Ver los ticket Asignar un ticket Salr del programa

codigo

```
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"ID:{self.id} | Tipo:{self.tipo} | Prioridad:{self.prioridad} | Estado:{self


# ------------------------
# Clase padre Empleado
class Empleado:
    def __init__(self, nombre):
        self.nombre = nombre

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


# ------------------------
# Clases hijas
class Desarrollador(Empleado):
    def trabajar_ticket(self, ticket):
        if ticket.tipo == "software":
            ticket.estado = "resuelto"
            print(f"[OK] El ticket {ticket.id} fue resuelto por {self.nombre}")
        else:
            ticket.estado = "no resuelto"
            print(f"[X] El ticket {ticket.id} no pudo ser resuelto por {self.nombre}")


class Tester(Empleado):
    def trabajar_ticket(self, ticket):
        if ticket.tipo == "prueba":
            ticket.estado = "resuelto"
            print(f"[OK] El ticket {ticket.id} fue resuelto por {self.nombre}")
        else:
```

```
            ticket.estado = "no resuelto"
            print(f"[X] El ticket {ticket.id} no pudo ser resuelto por {self.nombre}")


class ProjectManager(Empleado):
    def asignar_ticket(self, ticket, empleado):
        print(f"[Asignación] {self.nombre} asignó el ticket {ticket.id} al empleado {empleado
        empleado.trabajar_ticket(ticket)


# ------------------------
# Datos iniciales
tickets = []
empleados = [
    Desarrollador("Carlitos"),
    Tester("Juanillo"),
    ProjectManager("Marianita")
]

# ------------------------
# Simulación de menú sin input() para PDF
simulacion_opciones = [
    {"opcion": "1", "tipo": "software", "prioridad": "alta"},
    {"opcion": "1", "tipo": "prueba", "prioridad": "media"},
    {"opcion": "3", "ticket_id": 1, "empleado_index": 0},
    {"opcion": "3", "ticket_id": 2, "empleado_index": 1},
    {"opcion": "2"},
    {"opcion": "4"}
]

for accion in simulacion_opciones:
    if accion["opcion"] == "1":
        # Crear ticket
        id_ticket = len(tickets) + 1
        tipo = accion["tipo"]
        prioridad = accion["prioridad"]
        nuevo_ticket = Ticket(id_ticket, tipo, prioridad)
        tickets.append(nuevo_ticket)
        print(f"[Creación] Ticket {id_ticket} creado: Tipo={tipo}, Prioridad={prioridad}")

    elif accion["opcion"] == "2":
        # Ver tickets
```

```
        if not tickets:
            print("[Aviso] No hay tickets creados")
        else:
            print("\n--- LISTA DE TICKETS ---")
            for t in tickets:
                print(t)

    elif accion["opcion"] == "3":
        # Asignar ticket
        ticket_selec = next((t for t in tickets if t.id == accion["ticket_id"]), None)
        if ticket_selec:
            empleado_selec = empleados[accion["empleado_index"]]
            pm = next((e for e in empleados if isinstance(e, ProjectManager)), None)
            if pm:
                pm.asignar_ticket(ticket_selec, empleado_selec)
        else:
            print(f"[Error] Ticket {accion['ticket_id']} no encontrado")

    elif accion["opcion"] == "4":
        print("[Fin] Saliendo de la simulación...\n")
        break

print("\n--- ESTADO FINAL DE TICKETS ---")
for t in tickets:
    print(t)
```

```
[Creación] Ticket 1 creado: Tipo=software, Prioridad=alta
[Creación] Ticket 2 creado: Tipo=prueba, Prioridad=media
[Asignación] Marianita asignó el ticket 1 al empleado Carlitos
[OK] El ticket 1 fue resuelto por Carlitos
[Asignación] Marianita asignó el ticket 2 al empleado Juanillo
[OK] El ticket 2 fue resuelto por Juanillo

--- LISTA DE TICKETS ---
ID:1 | Tipo:software | Prioridad:alta | Estado:resuelto
ID:2 | Tipo:prueba | Prioridad:media | Estado:resuelto
[Fin] Saliendo de la simulación...


--- ESTADO FINAL DE TICKETS ---
ID:1 | Tipo:software | Prioridad:alta | Estado:resuelto
ID:2 | Tipo:prueba | Prioridad:media | Estado:resuelto
```