Instrucciones Generales

Para cada ejercicio, deberán crear al menos una **clase** que represente el objeto o concepto descrito. Piensen en sus **atributos** (variables de instancia) y **comportamientos** (métodos). Asegúrense de que las clases tengan **constructores** (al menos uno por defecto y uno con parámetros) y métodos `get` y `set` para sus atributos relevantes. Además, creen una clase `Main` (o similar) con un método `main` para probar sus clases, creando objetos y llamando a sus métodos. Agrega menues e interacción con el usuario en la consola o terminal, cuando lo creas necesario, para poder utilizar los objetos.

Java OOP - Sin Herencia

Ejercicio 1: Criaturas Fantásticas

Crea una clase `CriaturaMagica` con atributos como `nombre` (String), `tipoElemento` (String, ej., Fuego, Agua, Tierra), `nivelPoder` (int) y `estaDomesticada` (boolean). Implementa un método `lanzarHechizo()` que imprima un mensaje como [Nombre] lanza un hechizo de [TipoElemento]! y un método `intentarDomesticar()` que cambie el estado de `estaDomesticada`.

Ejercicio 2: Máquina de Vending Intergaláctica

Diseña una clase `ProductoVending` con atributos: `nombre` (String), `precioCreditos` (int) y `cantidadDisponible` (int). Agrega un método `comprar()` que disminuya la `cantidadDisponible` si hay stock y un método `reponer(cantidad)` para añadir más productos.

Ejercicio 3: Nave Espacial

Define una clase `NaveEspacial` con atributos `nombre` (String), `capacidadPasajeros` (int), `velocidadMaxima` (double, en warp), y `combustibleActual` (double). Crea un método `despegar()` que imprima La nave [Nombre] despega! y un método `reabastecer(cantidad)` que aumente el combustible.

Ejercicio 4: Recolector de Basura Espacial

Crea una clase `ChatarraEspacial` con atributos `tipo` (String, ej., Metal, Plástico, Orgánico), `pesoKG` (double) y `valorMonedas` (int). Implementa un constructor y un método `recolectar()` que simule la recolección, imprimiendo qué tipo de chatarra se recogió y su valor.

Ejercicio 5: Creador de Recetas de Pociones

Desarrolla una clase `Pocion` con atributos `nombre` (String), `color` (String), `efecto` (String) y `tiempoDuracionMinutos` (int). Implementa un método `mezclarIngredientes()` que imprima Mezclando ingredientes para la poción [Nombre]... y un método `beberPocion()` que imprima [Nombre] fue bebida. ¡El efecto es [Efecto] durante [TiempoDuracionMinutos] minutos!.

EJEMPLO: Imagina que estás en la cocina y decides hacer una poción saludable que te dé un impulso de energía. Para esta poción, definimos unas características básicas:

- **Nombre**: "Poción Pilas"
- **Color**: "Verde"
- **Efecto**: "Proporcionar energía"
- **Duración**: 60 minutos

Es como si estuvieras escribiendo una receta: "Voy a preparar una poción verde llamada Poción Pilas, y cuando lo beba, me va a dar energía durante una hora". Ahora, veamos cómo se prepara y qué pasa al tomarlo.

Paso 1: Mezclar los ingredientes
Para hacerlo, echas a la licuadora un puñado de espinacas, un plátano maduro, una manzana, un toque de jengibre, unas frambuesas de estrella y unos cubos de hielo. Prendes la licuadora y pasa esto:

"Mezclando ingredientes para la Poción Pilas..."

La licuadora zumba, y en unos segundos, tienes una mezcla suave y verde brillante. ¡Ya está lista para servir!

Paso 2: Beber la poción
Te sirves un vaso y lo tomas. Al probarlo, sientes el sabor fresco y natural, y entonces:

"Poción Pilas fue bebida. ¡El efecto es proporcionar energía durante 60 minutos!"

De repente, notas un subidón de vitalidad. Te sientes más despierto y listo para cualquier cosa, como si hubieras recargado tus pilas. Ya puedes irte a cazar goblins.

Ejercicio 6: Robot de Limpieza Doméstica

Crea una clase `RobotLimpiador` con atributos `nombre` (String), `nivelBateria` (int), `areaActual` (String, ej., Cocina, Sala) y `estaLimpiando` (boolean). Agrega métodos `cargarBateria()` (hasta 100), `limpiarHabitacion(nombreHabitacion)` que cambie `areaActual` y ponga `estaLimpiando` en true, y `detenerLimpieza()`.

Ejercicio 7: Coleccionista de Gemas Raras

Diseña una clase `Gema` con atributos `nombre` (String), `color` (String), `quilates` (double) y `pureza` (int, 1-100). Implementa un método `tallar()` que aumente la `pureza` y un método `vender(precioPorQuilate)` que calcule el valor de venta.

Ejercicio 8: Constructor de Ciudades Flotantes

Crea una clase `ModuloCiudad` con atributos `tipo` (String, ej., Residencial, Comercial, Energía), `energiaConsumida` (int) y `habitantesMaximos` (int). Agrega un método `expandir()` que simule la expansión y un método `generarReporte()` que muestre el tipo y el consumo de energía.

Ejercicio 9: Simulación de Eventos Climáticos Extremos

Desarrolla una clase `FenomenoNatural` con atributos `tipo` (String, ej., Tornado, Terremoto, Ola de Calor), `intensidad` (int, 1-10) y `duracionHoras` (int). Implementa un método `desencadenar()` que imprima un mensaje dramático sobre el fenómeno y un método `evaluarImpacto()` que estime su gravedad.

Ejercicio 10: Libro de Recetas Mágicas

Crea una clase `Conjuro` con atributos `nombre` (String), `efecto` (String), `manaRequerido` (int) y `nivelDificultad` (String, ej., Fácil, Medio, Difícil). Implementa un método `lanzar()` que imprima el efecto del conjuro y un método `aprender()` que imprima un mensaje de aprendizaje.

Ejercicio 11: Explorador de Mazmorras

Define una clase `ItemInventario` con atributos `nombre` (String), `tipo` (String, ej., Arma, Armadura, Poción), `peso` (double) y `valorOro` (int). Agrega un método `usar()` que imprima un mensaje sobre el uso del ítem y un método `descartar()`.

Ejemplo: El "Cuchillo Oxidado"
Supongamos que Alex está caminando por un pasillo oscuro y encuentra un "Cuchillo Oxidado" tirado junto a un esqueleto. Este cacharro tiene estas características:

- **Nombre**: "Cuchillo Oxidado"
- **Tipo**: "Arma"
- **Peso**: 1 kilogramo
- **Valor en oro**: 5 monedas

Es un cuchillo viejo y medio roto, pero algo es algo, ¿no? Alex lo recoge y lo mete en su mochila, pensando que podría servirle para defenderse.

Usando el Cuchillo Oxidado
Más adelante, Alex se topa con una rata gigante que le gruñe desde un rincón. Decide sacar el cuchillo y usarlo para espantarla o, si se pone fea la cosa, darle un buen pinchazo. Al blandirlo, imagina que una voz mágica (o su propia cabeza, quién sabe) le dice:

"Usando Cuchillo Oxidado: ¡El ítem está siendo utilizado según su tipo (Arma)!"

En palabras simples, Alex agarra el cuchillo, lo mueve como puede y logra ahuyentar a la rata. No es una espada legendaria, pero hace el trabajo.

Descartando el Cuchillo Oxidado
Un poco después, Alex encuentra un "Martillo de Guerra" en un cofre, mucho más pesado pero también más poderoso. Su mochila ya está hasta el tope, así que decide dejar el cuchillo viejo para llevarse el martillo. Al tirar el "Cuchillo Oxidado" al suelo, escucha (o se imagina) otro mensaje:

"Descartando Cuchillo Oxidado: El ítem ha sido dejado en la mazmorra."

Ejercicio 12: Fabrica de Juguetes Defectuosos

Crea una clase `Juguete` con atributos `nombre` (String), `material` (String), `defectuoso` (boolean) y `precio` (double). Implementa un método `inspeccionar()` que ponga `defectuoso` en true si cumple alguna condición (ej., material plástico y nombre Robot -¿defectuoso) y un método `reparar()` que ponga `defectuoso` en false.

Ejercicio 13: Simulador de Jardín Zen

Diseña una clase `ElementoZen` con atributos `tipo` (String, ej., Piedra, Arena, Agua), `posicionX` (int) y `posicionY` (int). Agrega un método `mover(nuevaX, nuevaY)` y un método `observar()` que imprima la posición del elemento.

Ejercicio 14: Tienda de Mascotas Virtuales

Crea una clase `MascotaVirtual` con atributos `nombre` (String), `especie` (String), `nivelHambre` (int, 0-100), y `nivelFelicidad` (int, 0-100). Implementa métodos `alimentar()` que disminuya `nivelHambre` y aumente `nivelFelicidad`, y `jugar()` que aumente `nivelFelicidad`.

Ejercicio 15: Sistema de Gestión de Bibliotecas de Música Retro

Desarrolla una clase `DiscoVinilo` con atributos `tituloAlbum` (String), `artista` (String), `genero` (String) y `estado` (String, ej., Excelente, Bueno, Rayado). Implementa un método `reproducirLadoA()` que simule la reproducción y un método `evaluarEstado()` que imprima el estado actual del vinilo.

Herencia

Instrucciones para Herencia

En estos ejercicios, deberán identificar una **clase base (superclase)** y una o más **clases derivadas (subclases)** que hereden de ella. Utilicen la palabra clave `extends`. Recuerden llamar al constructor de la superclase con `super()`.

Ejercicio 16: La Evolución de los Glitches Digitales

- Crea una clase base `GlitcheDigital` con atributos `tipo` (String, ej., Visual, Sonoro, Datos Corruptos) y `gravedad` (int, 1-10). Agrega un método `manifestarse()` que describa cómo se ve el glitch.
- Crea dos subclases: `GlitcheGrafico` y `GlitcheSonoro`.
- `GlitcheGrafico` debe tener un atributo adicional `colorDominante` (String). Sobrescribe el método `manifestarse()` para describir un glitch visual específico.
- `GlitcheSonoro` debe tener un atributo adicional `frecuenciaHz` (int). Sobrescribe el método `manifestarse()` para describir un glitch auditivo.

- En el `main`, crea objetos de `GlitcheGrafico` y `GlitcheSonoro` y llama a su método `manifestarse()`.

Ejercicio 17: Artefactos Perdidos de una Civilización Antigua

- Crea una clase base `ArtefactoAntiguo` con atributos `nombre` (String), `eraHistorica` (String) y `valorEstimado` (double). Agrega un método `examinar()` que describa el artefacto.
- Crea dos subclases: `JoyeriaAntigua` y `HerramientaMisteriosa`.
- `JoyeriaAntigua` puede tener un atributo adicional `materialPrincipal` (String). Sobrescribe el método `examinar()` para detallar la joyería.
- `HerramientaMisteriosa` puede tener un atributo adicional `funcionDesconocida` (String). Sobrescribe el método `examinar()` para describir su naturaleza enigmática.
- En el `main`, crea una `JoyeriaAntigua` y una `HerramientaMisteriosa` y llama a su método `examinar()`.

Ejercicio 18: La Tripulación de la Estación Espacial

- Crea una clase base `MiembroTripulacion` con atributos `nombre` (String), `rango` (String) y `anosExperiencia` (int).
- Crea una clase `CientificoEspacial` que herede de `MiembroTripulacion` y agregue un atributo `areaEspecializacion` (String).
- Crea una subclase `IngenieroNave` que herede de `MiembroTripulacion` y agregue un atributo `certificacionMantenimiento` (String).
- Implementa métodos `realizarTarea()` en cada clase para describir la tarea específica de su rol. En las subclases, puedes complementar el comportamiento de la superclase.
- En el `main`, crea objetos de `CientificoEspacial` y `IngenieroNave` y llama a sus métodos `realizarTarea()`.

Ejercicio 19: Los Peligros del Mundo de los Videojuegos

- Crea una clase base `EnemigoJuego` con atributos `nombre` (String), `salud` (int) y `danioAtaque` (int). Agrega métodos `recibirDanio(cantidad)` y `atacar(objetivo)` (simulando un ataque).
- Crea dos subclases: `Goblin` y `Dragon`.
- `Goblin` puede tener un atributo `tipoArma` (String). Sobrescribe `atacar()` para que el Goblin use su arma.
- `Dragon` puede tener un atributo `alientoElemento` (String, ej., Fuego, Hielo). Sobrescribe `atacar()` para que el Dragón use su aliento elemental.
- En el `main`, crea un `Goblin` y un `Dragon`, y prueba sus métodos de ataque y daño.

Ejercicio 20: Los Habitantes del Bosque Encantado

- Crea una clase base `SerMagico` con atributos `nombre` (String) y `longevidad` (int, en años). Agrega un método `emitirLuz()` (simulando algún tipo de aura) y un método `interactuar()` (que describa una interacción genérica).
- Crea dos subclases: `Hada` y `Ent`.
- `Hada` puede tener un atributo `colorAlas` (String). Sobrescribe `emitirLuz()` para describir el brillo de sus alas.
- `Ent` (árbol viviente) puede tener un atributo `especieArbol` (String). Sobrescribe `interactuar()` para describir una interacción más lenta y sabia.
- En el `main`, crea un `Hada` y un `Ent`, y llama a sus métodos `emitirLuz()` e `interactuar()`.