# PROGRAMACIÓN II
# SEGUNDA ENTREGA DEL TRABAJO PRÁCTICO
# MODELADO DE TADs Y DIAGRAMA

Integrantes del grupo: Lobos, Aaron Alexander
Muñoz, Franco Tadeo

Docentes: José Nores - Miguel Gabrielli

Comisión: 03

Cuatrimestre - Año: Segundo Cuatrimestre - 2025

Sistema HomeSolution - Trabajo Práctico Parte 2
Ampliación del Problema y Modificaciones al Diseño

Nuevos Requerimientos

Tipos de Empleados
- Empleados Contratados (originales): Cobran por hora
- Empleados de Planta Permanente: Cobran por día, tienen categoría (INICIAL, TÉCNICO, EXPERTO), medio día cuenta como día completo, bonus 2% sin retrasos

Gestión de Retrasos
- Registrar cantidad de retrasos por empleado
- Dos estrategias de asignación: primer disponible o por menor cantidad de retrasos

Tareas Dinámicas
- Agregar tareas a proyectos en curso (actualiza fechas)
- Estados de tarea: terminada/no terminada
- Liberar empleado al terminar tarea

Historial
- Mantener historial de empleados por proyecto después de finalizar
- Extensibilidad para futuros tipos de empleados

TADs Modificados y Nuevos

TADs Nuevos:
- Fecha (manejo de fechas y operaciones temporales)
- EmpleadoContratado (empleado que cobra por hora)
- EmpleadoPlanta (empleado que cobra por día)
- HistorialProyecto (registro de empleados que trabajaron)

TADs Modificados:
- Tarea (agrega estado terminada, referencia a empleado)
- Proyecto (usa Fecha, historial, agregar tareas dinámicamente)
- HomeSolution (maneja ambos tipos de empleados, estrategias asignación)

TAD Fecha (NUEVO)
Especificación
Valores (Atributos)
- fecha: LocalDate - Fecha interna (año, mes, día)

IREP
- fecha no puede ser null
- Debe representar una fecha válida del calendario

Estructuras de Datos
Utiliza LocalDate de Java que provee operaciones nativas para manejo de fechas.
Interfaz Pública

Operaciones
- crearFecha(anio: int, mes: int, dia: int): Fecha - Crea fecha con valores especificados
- crearFechaDesdeFormato(aaaammdd: int): Fecha - Crea fecha desde formato AAAAMMDD
- agregarDias(f: Fecha, dias: int): Fecha - Retorna nueva fecha sumando días especificados
- esAnterior(f1: Fecha, f2: Fecha): boolean - Verifica si f1 es anterior a f2
- esPosterior(f1: Fecha, f2: Fecha): boolean - Verifica si f1 es posterior a f2
- esIgual(f1: Fecha, f2: Fecha): boolean - Verifica si ambas fechas son iguales
- diasEntre(f1: Fecha, f2: Fecha): int - Calcula días de diferencia entre fechas
- aFormato(f: Fecha): int - Convierte fecha a formato AAAAMMDD

TAD Cliente (SIN CAMBIOS)

Especificación

Valores (Atributos)
- nombre: String - Nombre del cliente
- telefono: String - Teléfono de contacto
- email: String - Correo electrónico

IREP
- nombre, telefono y email no pueden ser strings vacíos
- telefono debe tener formato válido (solo números y caracteres permitidos como +, -, espacios)
- email debe contener formato válido (debe contener @ y dominio válido)

Estructuras de Datos
No requiere estructuras de datos adicionales.

Interfaz Pública

Operaciones
- crearCliente(nom: String, tel: String, email: String): Cliente - Crea un nuevo cliente con los datos proporcionados

TAD EmpleadoContratado

Especificación

Valores (Atributos)
- nombre: String - Nombre del empleado
- legajo: int - Número identificatorio único
- valorHora: double - Valor por hora de trabajo
- disponible: boolean - Indica si está disponible para asignación
- cantidadRetrasos: int - Contador de veces que tuvo retrasos

IREP
- legajo debe ser único en el sistema y mayor que 0

- valorHora debe ser mayor que 0
- nombre no puede ser vacío
- cantidadRetrasos debe ser mayor o igual que 0
- La disponibilidad debe reflejar correctamente el estado de asignación actual

Estructuras de Datos
No requiere estructuras de datos adicionales. Representa un empleado que cobra por hora.

Interfaz Pública
Operaciones
- crearEmpleadoContratado(nom: String, leg: int, valor: double): EmpleadoContratado - Crea empleado con disponibilidad true y cantidadRetrasos en 0
- estaDisponible(e: EmpleadoContratado): boolean - Indica si está disponible para asignación
- marcarComoAsignado(e: EmpleadoContratado): EmpleadoContratado - Marca como no disponible
- marcarComoDisponible(e: EmpleadoContratado): EmpleadoContratado - Marca como disponible
- incrementarRetrasos(e: EmpleadoContratado): EmpleadoContratado - Aumenta contador de retrasos en 1
- calcularPago(e: EmpleadoContratado, dias: double): double - Calcula pago según días trabajados (0.5 = 4h, ≥1 = 8h por día)
- tuvoRetrasos(e: EmpleadoContratado): boolean - Indica si tiene retrasos registrados (cantidad > 0)
- compararPorRetrasos(e1: EmpleadoContratado, e2: EmpleadoContratado): int - Compara cantidad de retrasos, retorna -1 si e1<e2, 0 si iguales, 1 si e1>e2

TAD EmpleadoPlanta
Especificación
Valores (Atributos)
- nombre: String - Nombre del empleado
- legajo: int - Número identificatorio único
- valorDia: double - Valor por día completo de trabajo
- categoria: String - Categoría del empleado (INICIAL, TÉCNICO, EXPERTO)
- disponible: boolean - Indica si está disponible para asignación
- cantidadRetrasos: int - Contador de veces que tuvo retrasos

IREP
- legajo debe ser único en el sistema y mayor que 0
- valorDia debe ser mayor que 0
- nombre no puede ser vacío
- categoria debe ser "INICIAL", "TÉCNICO" o "EXPERTO"
- cantidadRetrasos debe ser mayor o igual que 0
- La disponibilidad debe reflejar correctamente el estado de asignación actual

Estructuras de Datos
No requiere estructuras de datos adicionales. Representa un empleado de planta permanente.

Interfaz Pública

Operaciones
- crearEmpleadoPlanta(nom: String, leg: int, valorD: double, cat: String): EmpleadoPlanta - Crea empleado con disponibilidad true y cantidadRetrasos en 0
- estaDisponible(e: EmpleadoPlanta): boolean - Indica si está disponible para asignación
- marcarComoAsignado(e: EmpleadoPlanta): EmpleadoPlanta - Marca como no disponible
- marcarComoDisponible(e: EmpleadoPlanta): EmpleadoPlanta - Marca como disponible
- incrementarRetrasos(e: EmpleadoPlanta): EmpleadoPlanta - Aumenta contador de retrasos en 1
- calcularPago(e: EmpleadoPlanta, dias: double, sinRetrasos: boolean): double - Calcula pago (medio día = día completo), aplica bonus 2% si sinRetrasos es true
- tuvoRetrasos(e: EmpleadoPlanta): boolean - Indica si tiene retrasos registrados (cantidad > 0)
- compararPorRetrasos(e1: EmpleadoPlanta, e2: EmpleadoPlanta): int - Compara cantidad de retrasos, retorna -1 si e1<e2, 0 si iguales, 1 si e1>e2

TAD HistorialProyecto
Especificación
Valores (Atributos)
- numeroProyecto: int - Número del proyecto al que pertenece
- empleadosAsignados: HashSet<EmpleadoContratado | EmpleadoPlanta> - Conjunto de empleados que trabajaron
- tareasRealizadas: HashMap<EmpleadoContratado | EmpleadoPlanta, ArrayList<Tarea>> - Mapeo empleado → lista de tareas realizadas

IREP
- numeroProyecto debe ser mayor que 0 y corresponder a un proyecto existente
- Todos los empleados en empleadosAsignados deben haber trabajado efectivamente en el proyecto
- Para cada empleado en tareasRealizadas, debe estar también en empleadosAsignados
- Las tareas en tareasRealizadas deben pertenecer al proyecto

Estructuras de Datos
HashSet<Empleado>: Almacena referencias a empleados únicos que trabajaron. Permite verificación rápida O(1) de participación.
HashMap<Empleado, ArrayList<Tarea>>: Mapea cada empleado a las tareas que realizó. Permite consulta eficiente del historial detallado por empleado.

Interfaz Pública
Operaciones
- crearHistorial(numProy: int): HistorialProyecto - Inicializa historial vacío para un proyecto
- registrarEmpleadoEnTarea(h: HistorialProyecto, emp: EmpleadoContratado | EmpleadoPlanta, tarea: Tarea): HistorialProyecto - Agrega empleado y tarea al historial
- participoEmpleado(h: HistorialProyecto, emp: EmpleadoContratado | EmpleadoPlanta): boolean - Verifica si empleado trabajó en el proyecto
- cantidadEmpleados(h: HistorialProyecto): int - Devuelve cantidad de empleados que trabajaron

TAD Tarea (MODIFICADO)
Especificación
Valores (Atributos)
- titulo: String - Identificador de la tarea
- descripcion: String - Descripción detallada
- diasNecesarios: double - Días originalmente planificados
- diasRetraso: double - Días de retraso acumulados
- empleadoAsignado: EmpleadoContratado | EmpleadoPlanta - Empleado asignado (puede ser null)
- tieneAsignacion: boolean - Indica si tiene empleado asignado
- terminada: boolean - NUEVO: Indica si la tarea fue completada

IREP
- diasNecesarios debe ser mayor que 0, permite valores como 0.5 para medio día
- diasRetraso debe ser mayor o igual que 0
- Si tieneAsignacion es true, entonces empleadoAsignado no puede ser null
- Si tieneAsignacion es false, entonces empleadoAsignado debe ser null
- titulo no puede ser vacío
- descripcion no puede ser vacío
- Si terminada es true, entonces debe haber tenido asignación previamente

Estructuras de Datos
No requiere estructuras de datos adicionales. Almacena referencia al empleado asignado.

Interfaz Pública

Operaciones
- crearTarea(tit: String, desc: String, dias: double): Tarea - Crea tarea sin asignación y no terminada
- asignarEmpleado(t: Tarea, emp: EmpleadoContratado | EmpleadoPlanta): Tarea - Asigna empleado a la tarea
- registrarRetraso(t: Tarea, dias: double): Tarea - Suma días de retraso
- tieneEmpleadoAsignado(t: Tarea): boolean - Indica si tiene empleado asignado
- estaTerminada(t: Tarea): boolean - Indica si la tarea fue completada
- marcarComoTerminada(t: Tarea): Tarea - Marca tarea como terminada
- tuvoRetraso(t: Tarea): boolean - Indica si tiene retrasos acumulados (diasRetraso > 0)

TAD Proyecto (MODIFICADO)
Especificación
Valores (Atributos)
- numero: int - Identificador único
- cliente: Cliente - Datos del cliente
- direccion: String - Dirección de la vivienda
- tareas: ArrayList<Tarea> - Lista de tareas
- fechaInicio: Fecha - Fecha de inicio del proyecto
- fechaEstimadaFin: Fecha - Fecha estimada de finalización
- fechaRealFin: Fecha - Fecha real de finalización
- finalizado: boolean - Indica si está terminado

- costoCalculado: double - NUEVO: Costo precalculado al crear/modificar proyecto
- historial: HistorialProyecto - NUEVO: Registro de empleados que trabajaron

IREP
- numero debe ser único y mayor que 0
- fechaInicio ≤ fechaEstimadaFin ≤ fechaRealFin (usando operaciones de TAD Fecha)
- Si finalizado es true, todas las tareas deben estar terminadas
- tareas no puede estar vacía
- direccion no puede ser vacía
- costoCalculado debe ser mayor que 0
- Si una tarea está terminada, su empleado debe estar en el historial

Estructuras de Datos

ArrayList<Tarea>: Permite acceso O(1) por índice para asignación/reasignación, y agregar tareas dinámicamente al final.

HistorialProyecto: Mantiene registro de todos los empleados que trabajaron incluso después de finalizar el proyecto.

Interfaz Pública

Operaciones
- crearProyecto(num: int, cli: Cliente, dir: String, ts: ArrayList<Tarea>, fIni: Fecha, fEst: Fecha): Proyecto - Crea proyecto con historial vacío, costoCalculado y fechaRealFin igual a fechaEstimadaFin inicialmente
- agregarTarea(p: Proyecto, tit: String, desc: String, dias: double): Proyecto - Agrega tarea al final, actualiza fechaEstimadaFin y fechaRealFin sumando días, recalcula costo
- asignarEmpleadoATarea(p: Proyecto, indice: int, emp: EmpleadoContratado | EmpleadoPlanta): Proyecto - Asigna empleado a tarea y registra en historial
- registrarRetrasoEnTarea(p: Proyecto, indice: int, dias: double): Proyecto - Registra retraso en tarea, actualiza fechaRealFin sumando días y recalcula costo
- finalizarTarea(p: Proyecto, indice: int): Proyecto - Marca tarea como terminada (empleado ya está en historial)
- finalizar(p: Proyecto, fecha: Fecha): Proyecto - Marca proyecto como finalizado con fecha especificada
- estaFinalizado(p: Proyecto): boolean - Indica si el proyecto está terminado
- estaPendiente(p: Proyecto): boolean - Indica si hay tareas sin empleado asignado
- estaActivo(p: Proyecto): boolean - Indica si todas las tareas están asignadas pero no está finalizado
- hayRetrasos(p: Proyecto): boolean - Verifica si alguna tarea tiene retrasos acumulados
- cantidadTareas(p: Proyecto): int - Devuelve la cantidad total de tareas del proyecto

TAD HomeSolution (MODIFICADO)
Especificación
Valores (Atributos)

- empleadosContratados: HashMap<Integer, EmpleadoContratado> - Mapeo legajo → empleado contratado
- empleadosPlanta: HashMap<Integer, EmpleadoPlanta> - Mapeo legajo → empleado planta
- proyectos: HashMap<Integer, Proyecto> - Mapeo número → proyecto

IREP
- Todos los legajos en empleadosContratados y empleadosPlanta deben ser únicos entre sí y mayores que 0
- Un legajo no puede estar en ambos diccionarios de empleados
- Todos los números en proyectos deben ser únicos y mayores que 0
- Si un empleado no está disponible, debe tener al menos una tarea asignada en algún proyecto activo

Estructuras de Datos

HashMap<Integer, EmpleadoContratado>: Empleados que cobran por hora, acceso directo O(1) por legajo.

HashMap<Integer, EmpleadoPlanta>: Empleados de planta, acceso directo O(1) por legajo.

HashMap<Integer, Proyecto>: Proyectos con costo precalculado almacenado, consulta de costo O(1).

Interfaz Pública
Operaciones

Gestión de Empleados (Requerimiento 1)
- registrarEmpleadoContratado(hs: HomeSolution, nom: String, leg: int, valor: double): HomeSolution - Registra empleado contratado en el sistema (sin retrasos inicialmente)
- registrarEmpleadoPlanta(hs: HomeSolution, nom: String, leg: int, valorD: double, cat: String): HomeSolution - Registra empleado de planta con categoría especificada (sin retrasos inicialmente)

Gestión de Proyectos (Requerimientos 2, 6)
- registrarProyecto(hs: HomeSolution, num: int, cli: Cliente, dir: String, ts: ArrayList<Tarea>, fIni: Fecha, fEst: Fecha): HomeSolution - Registra proyecto con costo precalculado
- agregarTareaAProyecto(hs: HomeSolution, numProyecto: int, tit: String, desc: String, dias: double): HomeSolution - Agrega tarea a proyecto existente, actualiza fechas y recalcula costo

Asignación de Empleados (Requerimientos 3, 4)
- asignarEmpleadoTarea(hs: HomeSolution, numProyecto: int, indiceTarea: int): HomeSolution - Busca primer empleado disponible (cualquier tipo) y lo asigna
- asignarEmpleadoMenosRetrasos(hs: HomeSolution, numProyecto: int, indiceTarea: int): HomeSolution - Asigna empleado sin retrasos o con menor cantidad

Gestión de Tareas (Requerimientos 5, 7)

- registrarRetraso(hs: HomeSolution, numProyecto: int, indiceTarea: int, dias: double): HomeSolution - Registra retraso en tarea, incrementa contador del empleado asignado, actualiza fechas del proyecto y recalcula costo
- finalizarTarea(hs: HomeSolution, numProyecto: int, indiceTarea: int): HomeSolution - Marca tarea terminada, libera empleado (marca como disponible)

Finalización de Proyectos (Requerimiento 8)
- finalizarProyecto(hs: HomeSolution, numProyecto: int, fecha: Fecha): HomeSolution - Marca proyecto finalizado, libera todos los empleados asignados

Reasignación (Requerimientos 9, 10)
- reasignarEmpleado(hs: HomeSolution, numProyecto: int, indiceTarea: int, nuevoLegajo: int): HomeSolution - Cambia empleado asignado a tarea específica, libera anterior y asigna nuevo
- reasignarEmpleadoEficiente(hs: HomeSolution, numProyecto: int, indiceTarea: int): HomeSolution - Reasigna buscando empleado con menor cantidad de retrasos

Consultas de Costos (Requerimiento 11)
- consultarCostoProyecto(hs: HomeSolution, numProyecto: int): double - Devuelve costoCalculado del proyecto en O(1)

Informes de Proyectos (Requerimientos 12, 13, 14, 16)
- informarProyectosFinalizados(hs: HomeSolution): ArrayList<Integer> - Recorre proyectos y devuelve números de los que están finalizados
- informarProyectosPendientes(hs: HomeSolution): ArrayList<Integer> - Recorre proyectos y devuelve números de los que están pendientes
- informarProyectosActivos(hs: HomeSolution): ArrayList<Integer> - Recorre proyectos y devuelve números de los que están activos
- consultarEstadoProyecto(hs: HomeSolution, numProyecto: int): String - Devuelve "Finalizado", "Pendiente" o "Activo"

Consultas de Empleados (Requerimientos 15, 17, 18)
- informarEmpleadosNoAsignados(hs: HomeSolution): ArrayList<Integer> - Recorre empleados y devuelve legajos de los que están disponibles
- empleadoTuvoRetrasos(hs: HomeSolution, legajo: int): boolean - Busca empleado por legajo y verifica si tiene retrasos
- informarEmpleadosAsignadosProyecto(hs: HomeSolution, numProyecto: int): ArrayList<Integer> - Recorre tareas del proyecto y devuelve legajos de empleados asignados a tareas no terminadas
- informarHistorialEmpleadosProyecto(hs: HomeSolution, numProyecto: int): ArrayList<Integer> - Consulta historial del proyecto y devuelve legajos de todos los empleados que trabajaron

Justificación de Complejidades

O(1) para Consulta de Costo (Requerimiento 11):
El costo se calcula y almacena en proyecto.costoCalculado al:
1. Crear el proyecto

2. Agregar nueva tarea
3. Registrar retraso (puede cambiar margen de 35% a 25%)

La consulta es un acceso directo: proyectos[numProyecto].costoCalculado = O(1)

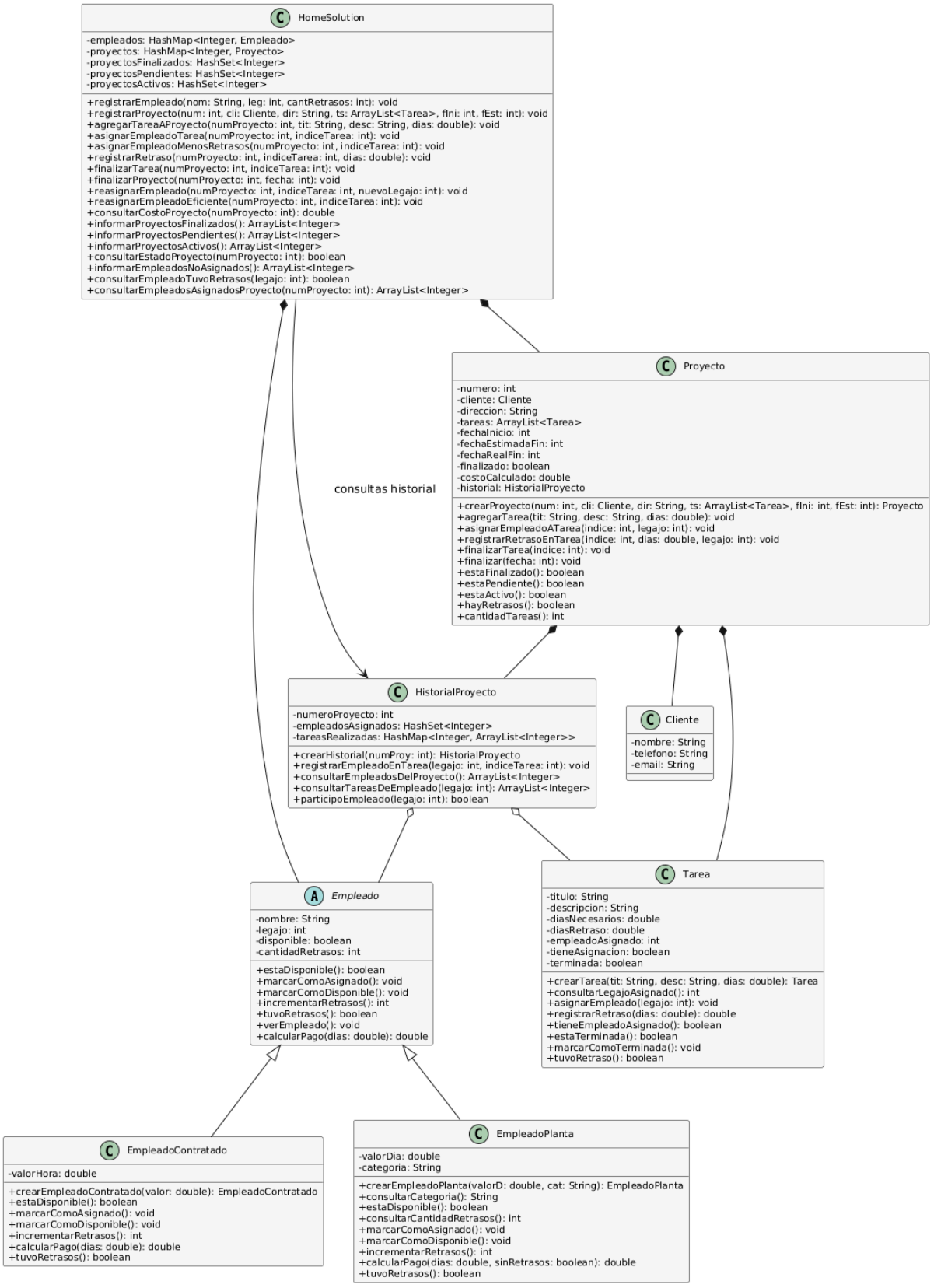
Reasignación (Requerimiento 9):
1. Acceso a proyecto: proyectos[numProyecto] → O(1)
2. Acceso a tarea: proyecto.tareas[indiceTarea] → O(1)
3. Obtener empleado actual de la tarea → O(1)
4. Marcar empleado actual como disponible → O(1)
5. Obtener nuevo empleado por legajo → O(1) con HashMap
6. Asignar nuevo empleado a tarea → O(1)
7. Marcar nuevo empleado como asignado → O(1)

Total: O(1)

Asignación por Menor Retrasos (Requerimientos 4, 10):
- Recorrer todos los empleados disponibles buscando el de menor retrasos
- Complejidad: O(n) donde n = cantidad total de empleados
- Es la forma más eficiente sin mantener estructuras ordenadas adicionales

Informes de Proyectos (Requerimientos 12, 13, 14):
- Recorrer HashMap de proyectos y filtrar por estado
- Complejidad: O(p) donde p = cantidad de proyectos
- No se mantienen conjuntos redundantes por simplicidad

# Diagrama de Clases



Relaciones del Diagrama:

- HomeSolution contiene EmpleadoContratado (1 a muchos)
- HomeSolution contiene EmpleadoPlanta (1 a muchos)
- HomeSolution contiene Proyecto (1 a muchos)
- Proyecto compone Cliente (1 a 1)
- Proyecto compone Tarea (1 a muchos)
- Proyecto compone HistorialProyecto (1 a 1)
- Proyecto usa Fecha (3 atributos de tipo Fecha)
- Tarea referencia EmpleadoContratado o EmpleadoPlanta (0..1 a 1)
- HistorialProyecto referencia EmpleadoContratado y EmpleadoPlanta (muchos a muchos)
- HistorialProyecto referencia Tarea (muchos a muchos)

Extensibilidad para Nuevos Tipos de Empleados

Estrategia Sin Herencia (solo con TADs):

Para agregar un nuevo tipo de empleado (ej: EmpleadoTemporal):

1. Crear nuevo TAD EmpleadoTemporal:
   - Define sus propios atributos específicos (ej: fechaFinContrato)
   - Implementa operaciones comunes: estaDisponible(), marcarComoAsignado(), etc.
   - Implementa su lógica de pago: calcularPago()

2. Modificar HomeSolution:
   - Agregar: HashMap<Integer, EmpleadoTemporal> empleadosTemporales
   - Agregar: registrarEmpleadoTemporal()
   - Modificar operaciones de asignación para considerar el nuevo tipo

3. Modificar Tarea:
   - El atributo empleadoAsignado puede ser cualquier tipo de empleado

4. Modificar HistorialProyecto:
   - Las estructuras ya soportan cualquier tipo de empleado

Impacto:
- Bajo: Solo se agrega storage en HomeSolution
- Localizado: No afecta TADs existentes
- Mantenible: Cada tipo mantiene su propia lógica de negocio
- NO Requiere: Modificar operaciones de búsqueda/asignación en HomeSolution

Resumen de Cambios Respecto a la Versión Anterior

Agregados:
1. TAD Fecha - Manejo apropiado de fechas con operaciones temporales
2. Uso de LocalDate - Implementación nativa para operaciones de fecha
3. Referencias a objetos - Tarea ahora guarda referencia a empleado, no legajo
4. HistorialProyecto con objetos - Guarda referencias a empleados y tareas reales

Eliminados:
1. Estructura asignacionesPorEmpleado - Redundancia innecesaria
2. Estructura asignacionesPorTarea - Redundancia innecesaria
3. Conjuntos proyectosFinalizados/Pendientes/Activos - Redundancia, se calculan recorriendo
4. Herencia - No existe en TADs, cada tipo es independiente
5. Parámetro cantidadRetrasos en constructores - Los empleados se crean sin retrasos

Modificados:
1. Proyecto usa TAD Fecha - En lugar de int para fechas
2. IREP más estricto - Sin redundancias, sin referencias a estructuras no declaradas
3. Complejidad O(1) real - Para costo, no para estados de proyecto
4. HomeSolution simplificado - Solo 3 estructuras esenciales
5. registrarEmpleadoPlanta - Ahora recibe categoría como parámetro

Notas de Implementación

Para la Parte 3 (Implementación):

1. Fecha: Se implementará usando LocalDate de Java
2. Polimorfismo: Se puede usar interfaces en implementación, pero no se modela en TADs
3. Búsqueda de empleados: Operaciones como asignarEmpleadoMenosRetrasos() recorrerán ambos HashMaps
4. Cálculo de estados: informarProyectosFinalizados() recorrerá proyectos verificando estaFinalizado()
5. Historial: Se mantiene sincronizado al asignar/finalizar tareas

Cumplimiento de Requerimientos

Req 1: Registrar empleado (contratado o planta con categoría)
Req 2: Registrar proyecto
Req 3: Asignar empleado a tarea (primer disponible)
Req 4: Asignar empleado con menos retrasos
Req 5: Registrar retraso en tarea
Req 6: Agregar tarea a proyecto existente
Req 7: Establecer tarea como finalizada
Req 8: Actualizar proyecto como finalizado
Req 9: Reasignar empleado en O(1)
Req 10: Reasignar empleado eficiente (menos retrasos)

Req 11: Informar costo en O(1)
Req 12: Informar proyectos finalizados
Req 13: Informar proyectos pendientes
Req 14: Informar proyectos activos
Req 15: Informar empleados no asignados
Req 16: Consultar si proyecto está finalizado
Req 17: Consultar si empleado tuvo retrasos
Req 18: Consultar empleados asignados a proyecto