Ejercicio integrador

Sistema carcelario

La dirección de una prisión busca modernizar su sistema de gestión para mejorar la seguridad y eficiencia en el manejo de los reclusos y las instalaciones. El nuevo sistema informático debe ser capaz de gestionar de manera ordenada y efectiva la información sobre los reclusos, el personal de la prisión y las distintas actividades dentro del recinto penitenciario.

Requisitos

Personal de la prisión

El sistema debe gestionar la información del personal de la prisión. Se deben registrar detalles como el nombre completo y el número de legajo.

Reclusos

El sistema debe ser capaz de gestionar la información de los reclusos, incluyendo detalles como el nombre completo y un número de identificación único dentro del sistema, que no puede ser modificado una vez asignado.

Los reclusos pueden pertenecer a diferentes categorías de seguridad según la gravedad de su delito. Por ejemplo, pueden ser de seguridad mínima, media o máxima. Según el nivel de seguridad del recluso, se le permitirá participar en diferentes actividades dentro del recinto penitenciario.

Actividades permitidas

Al recluso se le permite participar en diferentes actividades dentro del recinto penitenciario. Para cada nivel de seguridad, las actividades permitidas pueden variar según el grado de restricción y supervisión requerido.

Nivel de seguridad mínima

- Acceso a áreas comunes como el patio de recreo bajo supervisión.
- Participación en programas de educación y capacitación.
- Uso de biblioteca.
- Visita de familiares bajo supervisión.
- Participación en programas de rehabilitación.
- Tareas de mantenimiento bajo supervisión.

Nivel de seguridad media

- Participación en programas de rehabilitación.

- Tareas de mantenimiento bajo supervisión.

Nivel de seguridad máxima

- Participación en programas de rehabilitación.

Tareas

Se solicita diseñar e implementar un sistema que:

1. Organice la información de los reclusos, el personal de la prisión y las actividades en un formato fácilmente accesible y actualizado.
2. Permita realizar consultas y búsquedas rápidas sobre la información de los reclusos, el personal y las actividades.

Posible solución

Identificación de clases, atributos y métodos

1. Clase abstracta Recluso que implementa Permiso
    a. private static int identificador
    b. private int ID
    c. private String nombre
    d. private String Actividad actividadEnCurso
    e. Constructor con nombre. ID autoincremental
    f. Getter para ID y nombre
    g. Setter para nombre
    h. Método public void iniciarActividad(Actividad actividad) que establece que el recluso se encuentra actualmente en dicha actividad
    i. Método public void finalizarActividad() que establece que el recluso no se encuentra en ninguna actividad
    j. Método public Actividad informarActividadEnCurso() que devuelve la actividad en curso del recluso
    k. Método abstracto boolean puedeRealizarActividad(TipoActividad tipo) que determina si el recluso puede realizar la actividad según el tipo
2. Clase ReclusoSeguridadMinima que extiende Recluso
3. Clase ReclusoSeguridadMedia que extiende Recluso
4. Clase ReclusoSeguridadMaxima que extiende Recluso
5. Clase Personal
    a. private final String legajo
    b. private String nombre
    c. private boolean enSupervision
    d. Constructor con legajo y nombre
    e. Getter para legajo y nombre
    f. Setter para nombre
    g. Método public void supervisar() que establece que el personal está asignado a una actividad
    h. Método public void liberar() que establece que el personal se liberó de una actividad
6. Clase Actividad
    a. private Recluso recluso
    b. private LocalDate fechaInicio
    c. private LocalDate fechaFin
    d. private TipoActividad tipo
    e. Constructor con recluso y tipo
    f. Getter para todos los atributos
    g. Método public void finalizar() que establece la fecha de fin de la actividad
7. Clase ActividadSupervisable que extiende Actividad e implementa Supervisable
    a. private Personal supervisor
    b. Getter para supervisor
8. Enumeración TipoActividad
    a. PATIO(“Actividad en el patio”)
    b. PROGRAMA_EDUCACION (“Programa educativo”)

    c. BIBLIOTECA (“Consulta en biblioteca”)
    d. VISITA_FAMILIAR (“Visita familiar”)
    e. PROGRAMA_REHABILITACION (“Programa de rehabilitación”)
    f. TAREA_MANTENIMIENTO (“Tarea de mantenimiento”)
9. Interfaz Supervisable
    a. Método void iniciarSupervision(Personal supervisor) que inicia la actividad, dejando al personal no disponible para otra actividad y asignando una actividad en curso al recluso
    b. Método void cambiarSupervisor(Personal nuevo) que cambia el supervisor asignado a la actividad
    c. Método void finalizar() que finaliza la actividad, liberando al personal para supervisar otra actividad e indicando que el recluso no está en ninguna actividad específica
10. Clase GestorReclusos
    a. Listado privado de todos los reclusos de la prisión
    b. Método public Recluso buscarRecluso(int identificador) que devuelve la referencia al recluso que tiene el identificador pasado por parámetro o null si no lo encuentra
    c. Método public Recluso buscarRecluso(String nombre) que devuelve la referencia al recluso que tiene el nombre pasado por parámetro o null si no lo encuentra
    d. Método public void listarReclusosEnActividad() que lista la información de todos los reclusos que se encuentran en una actividad
    e. Método public Recluso crearRecluso(String nombre, NivelSeguridad nivel) que crea un recluso según el nivel de seguridad asignado
11. Clase GestorPersonal
    a. Listado privado de todo el personal de la prisión
    b. Método public Personal buscarPersonal(String legajo) que devuelve la referencia al personal que tiene el legajo pasado por parámetro o null si no lo encuentra
    c. Método public Personal crearPersonal(String legajo, String nombre) que crea un nuevo personal
    d. Método public Personal buscarPersonalDisponible() que devuelve el primer personal que encuentra disponible para supervisar una actividad
12. Clase GestorActividades
    a. Listado privado de todas las actividades pasadas y en curso de la prisión
    b. Método public void listarActividadesEnCurso() que lista la información de todas las actividades que se encuentran en curso
    c. Método public void listarActividades(Recluso recluso, LocalDate fecha) que lista todas las actividades que realizó el recluso pasado por parámetro en cierta fecha
    d. Método public void listarActividades(Personal personal, LocalDate fecha) que lista todas las actividades que supervisó el personal pasado por parámetro en cierta fecha
    e. Método public boolean iniciarActividad(Recluso recluso, TipoActividad tipo, GestorPersonal gestor) que inicia una actividad de cierto tipo para el recluso y devuelve un booleano indicando si pudo realizar la acción. Si la actividad es

supervisable, le asigna un supervisor disponible. Valida que el usuario pueda realizar la actividad.

f. Método public void finalizarActividad(Recluso recluso) que finaliza la actividad en curso del recluso y libera al supervisor en caso de que sea una actividad supervisable

13. Enumeración NivelSeguridad
    a. MINIMO
    b. MEDIO
    c. MAXIMO

PlantUML

```
@startuml
!theme plain
abstract class Recluso implements Permiso {
    -static identificador: int
    -ID: int
    -nombre: String
    -Actividad actividadEnCurso
    +Recluso(nombre: String)
    +getID(): int
    +getNombre(): String
    +setNombre(nombre: String): void
    +iniciarActividad(actividad: Actividad): void
    +finalizarActividad(): void
    +informarActividadEnCurso(): Actividad
}
class ReclusoSeguridadMinima extends Recluso
class ReclusoSeguridadMedia extends Recluso
class ReclusoSeguridadMaxima extends Recluso
interface Permiso {
    +puedeSalirPatio(): boolean
    +puedeRealizarProgramaEducativo(): boolean
    +puedeConsultarBiblioteca(): boolean
    +puedeRecibirVisita(): boolean
    +puedeRealizarProgramaRehabilitacion(): boolean
    +puedeRealizarTareaMantenimiento(): boolean
}
class Personal {
    -final legajo: String
    -nombre: String
    -enSupervision: boolean
    +Personal(legajo: String, nombre: String)
    +getLegajo(): String
    +getNombre(): String
    +estaEnSupervision(): boolean
    +setNombre(nombre: String): void
    +supervisar(): void
    +liberar(): void
}
class Actividad {
    -recluso: Recluso
    -fechaInicio: LocalDate
    -fechaFin: LocalDate
    -tipo: TipoActividad
    +Actividad(recluso: Recluso, tipo: TipoActividad)
    +getRecluso(): Recluso
    +getFechaInicio(): LocalDate
    +getFechaFin(): LocalDate
    +getTipo(): TipoActividad
    +finalizar(): void
```

```
}
class ActividadSupervisable extends Actividad implements Supervisable {
    -supervisor: Personal
    +getSupervisor(): Personal
}
enum TipoActividad {
    PATIO("Actividad en el patio"),
    PROGRAMA_EDUCACION("Programa educativo"),
    BIBLIOTECA("Consulta en biblioteca"),
    VISITA_FAMILIAR("Visita familiar"),
    PROGRAMA_REHABILITACION("Programa de rehabilitación"),
    TAREA_MANTENIMIENTO("Tarea de mantenimiento")
}
interface Supervisable {
    +asignarSupervisor(supervisor: Personal): void
    +cambiarSupervisor(nuevo: Personal): void
    +iniciar(): void
    +finalizar(): void
}
class GestorReclusos {
    -reclusos: List<Recluso>
    +buscarRecluso(identificador: int): Recluso
    +buscarRecluso(nombre: String): Recluso
    +listarReclusosEnActividad(): void
    +crearRecluso(nombre: String, nivel: NivelSeguridad): Recluso
}
class GestorPersonal {
    -personal: List<Personal>
    +buscarPersonal(legajo: String): Personal
    +crearPersonal(legajo: String, nombre: String): Personal
    +buscarPersonalDisponible(): Personal
}
class GestorActividades {
    -actividades: List<Actividad>
    +listarActividadesEnCurso(): void
    +listarActividades(recluso: Recluso, fecha: LocalDate): void
    +listarActividades(personal: Personal, fecha: LocalDate): void
    +iniciarActividad(recluso: Recluso, tipo: TipoActividad, gestor:
GestorPersonal): boolean
    +finalizarActividad(recluso: Recluso): void
}
enum NivelSeguridad {
    MINIMO, MEDIO, MAXIMO
}
GestorReclusos *-- "1..*" Recluso : gestiona
GestorActividades *-- "1..*" Actividad : gestiona
GestorPersonal *-- "1..*" Personal : gestiona
ActividadSupervisable o-- "1" Personal : supervisa
Actividad o-- "1" TipoActividad : tiene
GestorReclusos ..> NivelSeguridad : necesita
@enduml
```

Diagrama UML