Paradigmas de Programación **Examen Final** **14/12/2024**

Condiciones de aprobación

Para aprobar es necesario simultáneamente: • completar el 60% del examen, y • obtener al menos la mitad de los puntos **en cada paradigma**.	**En todas tus respuestas sé puntual**, no pierdas el foco de lo que se pregunta. Respuestas en exceso generales son tan malas como respuestas incompletas. 

Parte A

Se necesita implementar un sistema para que los atletas se inscriban a las competencias de un torneo deportivo. Sabemos que un atleta puede inscribirse en una competencia cuando:

1. Ha completado la inscripción general al torneo.
2. No ha participado todavía en esa competencia.
3. Ha participado en todas las competencias requeridas como condición previa para esa competencia, a menos que la inscripción general al torneo se haya realizado hace menos de un año calendario, en cuyo caso las competencias previas no son requeridas.

Se propuso la siguiente solución:

```
puedeInscribirse(Atleta, Competencia, Fecha):-
  inscripcionGeneral(Atleta, FechaInscripcion),
  not(participo(Atleta, Competencia, _)),
  añosCalendarioTranscurridos(FechaInscripcion, Fecha, 0).

puedeInscribirse(Atleta, Competencia, _):-
  inscripcionGeneral(Atleta, _),
  not(participo(Atleta, Competencia, _)),
  competenciaRequerida(CompetenciaPrevia, Competencia),
  participo(Atleta, CompetenciaPrevia, _).
```

Se sabe que existen los siguientes predicados:

- `inscripcionGeneral/2`: Relaciona un atleta y la fecha en la que completó su inscripción general al torneo. Es inversible.
- `participo/3`: Relaciona un atleta, una competencia, y la fecha en la que participó. Es inversible.
- `competenciaRequerida/2`: Relaciona dos competencias tales que la primera es condición previa para la segunda. Es inversible.
- `añosCalendarioTranscurridos/3`: Relaciona dos fechas cualesquiera con los años calendario que transcurrieron entre ambas. Solo es inversible para su tercer parámetro.

1. ¿La solución propuesta cumple con la lógica pedida? Justifique y plantee algún ejemplo que sirva para fundamentar esa respuesta.
2. Analice la inversibilidad de `puedeInscribirse/3`. En caso de que no sea inversible para uno o más parámetros, explique qué sería necesario modificar o agregar para que lo sea.
3. Realice cualquier corrección que considere necesaria sobre `puedeInscribirse/3`, considerando las respuestas anteriores y eliminando cualquier repetición de lógica existente.

Parte B

Se desea modelar en el paradigma funcional un sistema de admisión para competencias deportivas. Existen atletas que desean participar en una competencia que tiene una serie de requisitos, de manera que no se permite la inscripción de aquellos que incumplan alguno de ellos. Puede haber diversos requisitos, como por ejemplo que solo se inscriban atletas de una cierta nacionalidad, que no se permita la inscripción de quienes tengan menos de cierta edad, o que no puedan participar atletas con ciertos tipos de equipamiento no permitido. Puede haber competencias con más requisitos que otras, sin requisitos, o con requisitos similares, como restringir la participación por nacionalidad, pero con diferentes criterios específicos. Por lo tanto, de los atletas se debe conocer la edad, la nacionalidad, y el equipamiento que llevan consigo.

1. Definir tipos de datos y funciones (explicitando el tipo de todas ellas) para cubrir las necesidades explicadas.
2. Mostrar cómo se representa una competencia de ejemplo que tenga tres requisitos como los mencionados

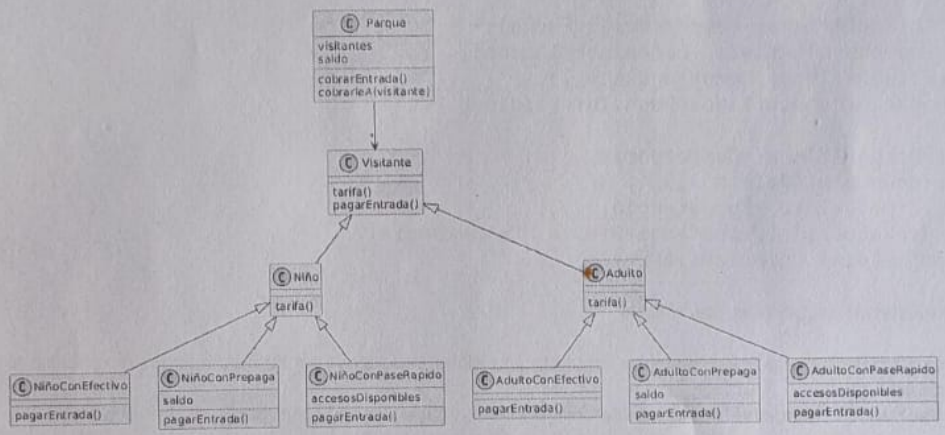
anteriormente.

3. Desarrollar la función `controlDeInscripcion`, que permita saber qué atletas de una lista de espera cumplen los requisitos para participar en la competencia.
4. Indicar dónde y para qué se utilizaron los siguientes conceptos: **Composición**, **Aplicación parcial**, **Orden superior**.

Parte C

Se sabe que en un parque de diversiones se cobra entrada a los visitantes. Los adultos y los niños tienen diferentes formas de calcular la tarifa (los adultos pagan una tarifa única de $100, mientras que los niños pagan $50), y además hay visitantes que pagan con "Pase Rápido", otros con efectivo y otros con tarjeta prepaga. Al efectuarse el cobro suceden dos cosas: el saldo del parque aumenta, y cada visitante efectúa el pago. Los que pagan con tarjeta prepaga disminuyen el saldo de su tarjeta en un 10% adicional como gasto de servicio (el 10% no va para el parque), y los que pagan con Pase Rápido disminuyen en 1 la cantidad de accesos disponibles. No necesitamos registrar nada adicional para los que pagan en efectivo.

Se tiene el siguiente diagrama de la solución propuesta:




```
class Parque {
  const property visitantes = []
  var property saldo = 0
  method cobrarEntrada() {
    visitantes.forEach { v => self.cobrarleA(v) }
  }
  method cobrarleA(visitante) {
    saldo = saldo + visitante.tarifa()
    visitante.pagarEntrada()
  }
}
```

1. **Indicar verdadero o falso y justificar en cada caso:**
   a. Para que la solución propuesta funcione, es necesario que exista la clase `Visitante`, ya que de lo contrario no podrían incluirse los distintos visitantes en la lista del parque.
   b. Hay buen uso de polimorfismo en la solución.
   c. Por cómo se planteó el modelo, se va a repetir lógica en las implementaciones de `pagarEntrada()`.
   d. Si un adulto que paga con tarjeta prepaga decide pasarse al sistema de Pase Rápido, podrá hacerlo sin problema con este modelo.
   e. Al cobrarle a un visitante que paga con Pase Rápido y este se queda sin accesos disponibles, se lanza un error, se frena la ejecución y el estado del sistema queda coherente.
2. Desarrollar una nueva solución que resuelva los problemas identificados. Debe incluir el código y un diagrama de clases actualizado.
3. ¿Qué haría falta hacer en cada solución para poder contemplar otros tipos de visitantes, como personas mayores o adolescentes?