


Condiciones de aprobación:

Para aprobar es necesario simultáneamente: <ul style="list-style-type: none"> • obtener 8 puntos de 14, y • obtener al menos la mitad de los puntos en cada paradigma. Las preguntas choice o V/F: <ul style="list-style-type: none"> • no serán consideradas si no están justificadas, y • se justifican mediante explicaciones y/o código a criterio del alumno 	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 tiene la siguiente función:

```
f x y = (>10).head.filter (x y).map y
```

1. Inferir el tipo de la función f
2. Mostrar un ejemplo de uso de la función f usando una lista infinita e indicar si dicha evaluación puede terminar justificando la respuesta conceptualmente.
3. V/F: Si f se definiera recursivamente en vez de usando `filter` y `map`, la solución sería más declarativa. ¿Por qué?

Parte B

Tenemos el siguiente código Prolog para armar un programa que ayude a organizar la cursada:

% relaciona una materia con el año % en el que se cursa <pre>materia(troncal(ads), 2). materia(electiva(tadp, 5), 3). materia(electiva(rrhh, 3), 3). materia(obligatoria(pdep), 2).</pre>	<pre>%relaciona dos nombres de materias tal que % la segunda es correlativa de la primera correlativa(tadp, pdep). correlativa(NombreElectiva, NombreTroncal):- materia(troncal(NombreTroncal), Anio), Anio is Anio + 1, materia(electiva(NombreElectiva, _), Anio).</pre>
---	--

1. La solución propuesta para el predicado `correlativa/2` tiene un problema. Indicar cuál es justificando conceptualmente por qué no es correcto.
2. Se agrega el siguiente predicado a nuestro programa:

```
habilitan(NombresMateriasAprobadas, NombreMateria):-
    forall(correlativa(NombreMateria, NombreAprobada),
        member(NombreAprobada, NombresMateriasAprobadas)).
```

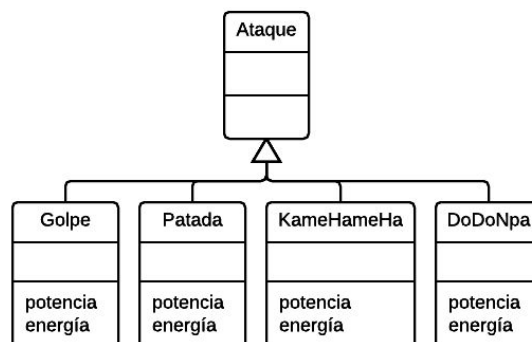
Asumiendo que los problemas del predicado `correlativa/2` son arreglados, indicar V/F para cada afirmación y justificar.

- a. `habilitan/2` no es inversible respecto a su primer parámetro. Lo correcto sería armar una lista con todas las materias para que pueda usarse de forma existencial.
- b. `habilitan/2` es inversible respecto a su segundo parámetro, porque `correlativa/2` es inversible.
- c. `habilitan/2` no es inversible respecto a su segundo parámetro, y convendría tratar polimórficamente a las materias para generar dicha variable.

Parte C

En el universo de Dragon Ball los guerreros emplean distintas técnicas de ataque como golpes de puño, patadas y ataques de ki orientado como el Kame Hame Ha y el Dodonpa. Cada una de estas técnicas de ataque consume una cantidad de energía y tiene una potencia que se calculan de formas distintas, que ya se encuentran resultados.

A. Para implementar la solución, un ingeniero en sistemas hizo el siguiente diagrama de clases, en el que a las cuatro clases que representan las técnicas de ataques, añadió la clase `Ataque`. Lo justificó de esta manera:



- Es una clase abstracta, porque no implementa métodos.
- Es necesaria en la solución para garantizar el polimorfismo.
- Permite organizar mejor la solución haciéndola más declarativa.

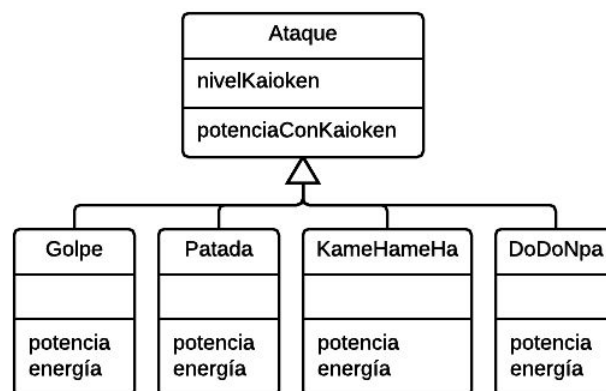
¿Qué opina de estas afirmaciones?

B. Hay un nuevo requerimiento, por el que se puede atacar mediante lo que se denomina Kaio Ken, que asociado a una técnica de ataque, aumenta su potencia. Cuando se ataca con un Kaio Ken, el mismo hace aumentar la potencia del ataque N veces, donde N es el nivel propio de cada Kaio Ken. Debe ser posible potenciar cualquier tipo de ataque y también usar ataques sin potenciar. Por ejemplo, puede haber patadas que se potencian con Kaio Ken y patadas que no, puede haber un Kaio Ken de nivel 10 que asociado a un golpe brinda una potencia 10 veces superior a la del golpe solo, y a la vez, se puede atacar con un golpe de manera aislada. Dos ingenieros implementaron este agregado de formas diferentes:

Solución 1

```
Wollok
#Ataque
method potenciaConKaioKen() {
  if (nivelKaioKen == null){
    return self.potencia() }
  return nivelKaioKen * self.potencia() }
```

```
Smalltalk
#Ataque
potenciaConKaioKen
^nivelKaioKen
ifNil: [self potencia]
ifNotNil: [nivelKaioKen * self potencia]
```

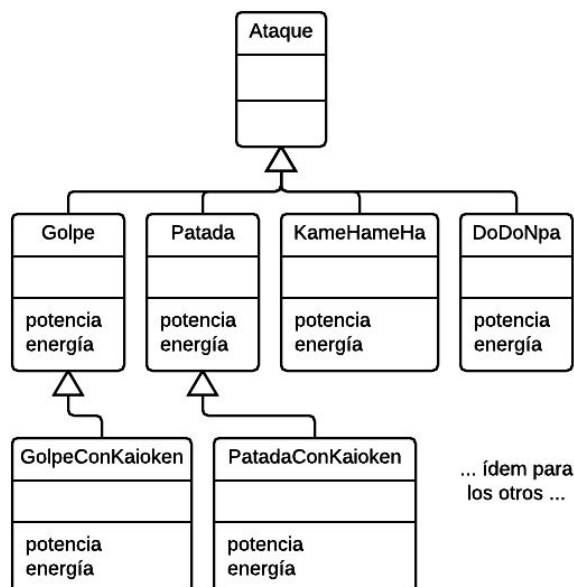


Solución 2

```
Wollok
#GolpeConKaioKen
method potencia() {
  return nivelKaioKen * super()
}
#PatadaConKaioKen
method potencia() {
  return nivelKaioKen * super()
}
```

```
Smalltalk
#GolpeConKaioKen
potencia
^nivelKaioKen * super potencia
#PatadaConKaioKen
potencia
^nivelKaioKen * super potencia
```

y así para los otros tipos de ataques...



B1. ¿Qué puntos fuertes y débiles encuentra en cada uno de estas soluciones? Justificar conceptualmente.

B2. Proponer una solución (incluyendo código y diagrama de clases) que incluya los nuevos requerimientos y resuelva los puntos débiles antes identificados.

B3. Agregar a la nueva solución un test de unidad donde se verifique la creación de un ataque KameHameHa potenciado por un KaioKen de nivel 10 y una patada no potenciada.