Parte A:

1. podrianAyudar problema personas = filter(any(sirve problema)).(map habilidades) personas

   Acá pifié feo. No devuelve una lista de personas por el map. Creo que lo correcto sería esto (chequear por las dudas):

   podrianAyudar problema personas = filter(/persona -> any(sirve problema) (habilidades persona)) personas

2. Se utiliza orden superior con las funciones “filter”, “any” y “map”, para convertir la lista en una lista de listas de habilidades, y luego filtrar las que contienen al menos una que sirve para el problema.

   Esto me lo pusieron como bien, supongo que porque, si bien la función estaba mal, supe explicar bien lo que hace conceptualmente.

   Se utiliza composición entre map y filter, para usarlos en ese respectivo orden, y que se devuelvan las personas indicadas.

   Acá no pusieron nada, pero asumo que está mal, porque la función no hace eso.

   Se usa aplicación parcial en el any, dejando como único parámetro la función sirve, para que luego reciba la lista de habilidades y filtre.

   Esto me lo corrigieron como bien, asumo que por la misma razón que lo primero.

3. En caso de poseer una habilidad que sirva, el any cortaría la recursividad y la persona pasaría el filter, permitiendo a la función compilar correctamente. En caso de no poseer ninguna, la función no compilaría, ya que tanto filter como any evalúan la lista entera, y no habría corte de flujo.

   La recursividad: Esto me lo tacharon. Se dice flujo y no recursividad.
   Compilar: Se dice ejecutar.
   Compilaría: sería terminaría.
   Evalúan la lista entera: “Por lazy evaluation no evalúan la lista entera, sino que intenta evaluar hasta donde se necesita”.

   Creo que me lo contaron como medio ejercicio bien. No estoy seguro.

Parte B:

1. tiene(persona1, CosaValiosa),
   tiene(persona2, OtraCosa),
   vale(CosaValiosa, ValorCosaValiosa),
   vale(OtraCosa, OtroValor)

2. a y b
   a. Esto no es posible, debido a que el predicado no es inversible. Para que lo sea, debemos generar a las personas:
      tiene(Persona1, _),
      tiene(Persona2, _),
      … resto de la función …

   b. not(todoLoQueTieneEsMasValioso(_, pedro)).

Esto está todo bien.

Parte C:

1. vof
   a. FALSO. Si bien se solucionaría parcialmente, todavía habría lógica repetida a la hora de evaluar si los personajes son actuados y si son rescatados.
   No hay lógica repetida con eso.

   b. Verdadero. Se está rompiendo su encapsulamiento a la hora de evaluar sus géneros de forma explícita en objetos ajenos.
   c. FALSO. Si bien no hay ifs, se utiliza mal el polimorfismo al evaluar explícitamente rompeEstereotipos para cada personaje.

En todos me pusieron un tic y 0,5. No se si significa que solo valió la mitad de cada punto, o si estaba todo bien y los puntos valen en total 0,5.

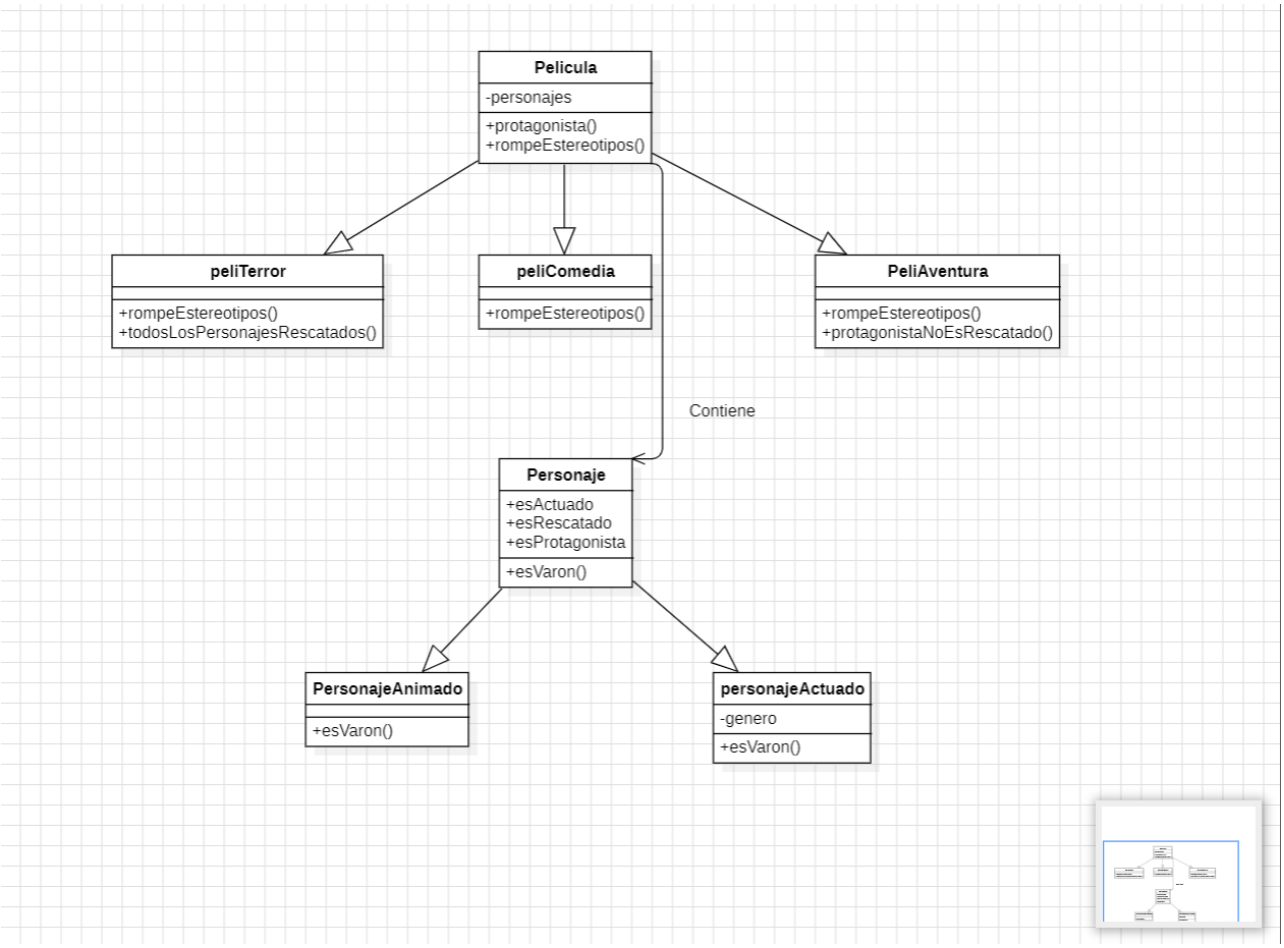
2. Código.

*resolucionfinal.wlk

```
class Personaje {
    var property esActuado
    var property esRescatado
    var property esProtagonista

    method esVaron()

}

class PersonajeAnimado inherits Personaje{
    override method esActuado() = false // No es necesario. Me pusieron como mal esto y todo lo que usa esActuado
    override method esRescatado() = false /* Estos dos métodos los escribí mal en el parcial, simplemente puse
                                             esActuado = false y esRescatado = false, pero no me lo tuvieron en cuenta*/
    override method esVaron() = false
}

class PersonajeActuado inherits Personaje{
    var genero

    override method esActuado() = true //También lo escribí mal en el parcial, tampoco me lo tuvieron en cuenta

    override method esVaron(){
        return genero == "masculino"
    }
}

class Pelicula{
    var personajes = #{}

    method protagonista() = personajes.find({p => p.esProtagonista()})

    method rompeEstereotipos(){
        return self.protagonista().esVaron().negate()
    }
}

```

```
class Pelicula{
    var personajes = #{}

    method protagonista() = personajes.find({p => p.esProtagonista()})

    method rompeEstereotipos(){
        return self.protagonista().esVaron().negate()
    }
}

class PeliAventura inherits Pelicula{
    override method rompeEstereotipos(){
        return super() and self.protagonistaNoEsRescatado() // en el parcial puse self.super() para todos los super. Re gagá jsjsj
    }

    method protagonistaNoEsRescatado(){
        return self.protagonista().esRescatado().negate()
    }
}

class PeliTerror inherits Pelicula{
    override method rompeEstereotipos(){
        return super() and self.todosLosPersonajesRescatados()
    }

    method todosLosPersonajesRescatados(){
        return personajes.all({p=>p.esRescatado()}) //Aca me pusieron una correccion que no entendí: "super() sud."
    }
}

class PeliComedia inherits Pelicula{
    override method rompeEstereotipos(){
        return super() and personajes.size() == 1
    }
}
```

Diagrama:



El diagrama está bien.

Mi nota final fue un 9.