PRÁCTICA 3

APARTADO A. CÓDIGO DE ANDAMIAJE

Hemos realizado una reificación de la clase Adopción.

Clase Adopción

```
import java.util.Date;


public class Adopcion {
    private Date fecha;
    private Animal animal;
    private Adoptante adoptante;

    public Adopcion(Date fecha, Animal animal, Adoptante adoptante) {
        this.setFecha(fecha);
        this.setAnimal(animal);
        this.setAdoptante(adoptante);
        this.animal.setAdopcion(this);
    }

    public Date getFecha() {
        return fecha;
    }

    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }

    public Animal getAnimal() {
        return animal;
    }

    public void setAnimal(Animal animal) {
        this.animal = animal;
    }
```

```
    public Adoptante getAdoptante() {
        return adoptante;
    }

    public void setAdoptante(Adoptante adoptante) {
        this.adoptante = adoptante;
    }
}
```

Clase Adoptante

```
public class Adoptante extends Socio {
    private List<Adopcion> adopciones;
    public Adoptante(Date registro,Refugio refugio) {
        super(registro,refugio);
        adopciones = new ArrayList<>();
    }

    public void adoptar(Animal a, Voluntario v) {
        v.tramitarAdopcion(a,this);
    }

    public List<Adopcion> getAdopciones() {
        return adopciones;
    }

    public void setAdopciones(List<Adopcion> adopciones) {
        this.adopciones = adopciones;
    }

    protected void añadirAdopcion(Adopcion ad) {
        adopciones.add(ad);
    }
```

Clase Animal

```
public class Animal {

	private Date nacimiento;

	private EstadoAnimal estado;

	private Adopcion adopcion;


	public Animal(Date nacimiento, EstadoAnimal estado) {

	this.setNacimiento(nacimiento);

	this.setEstado(estado);

	}


	public Date getNacimiento() {

	return nacimiento;

	}


	public void setNacimiento(Date nacimiento) {

	this.nacimiento = nacimiento;

	}


	public EstadoAnimal getEstado() {

	return estado;

	}


	public void setEstado(EstadoAnimal estado) {

	this.estado = estado;

	}


	public void setAdopcion(Adopcion adopcion) {

	this.adopcion = adopcion;

	}


	public Adopcion getAdopcion() {
```

```
        return adopcion;
        }
}
```

Clase Donación

```
public class Donacion {
        private Float cantidad;
        private Date fecha;

        public Donacion(Float cantidad, Date fecha) {
        this.setCantidad(cantidad);
        this.setFecha(fecha);
        }

        public Float getCantidad() {
        return cantidad;
        }

        public void setCantidad(Float cantidad) {
        this.cantidad = cantidad;
        }

        public Date getFecha() {
        return fecha;
        }

        public void setFecha(Date fecha) {
        this.fecha = fecha;
        }
}
```

Clase Donante

```
public class Donante extends Socio {
        ArrayList<Donacion> donaciones;
```

```
	public Donante(Date registro, Refugio refugio) {
	super(registro, refugio);
	donaciones = new ArrayList<>();
	}

	public void donar(Float c) {
	Donacion d = new Donacion(c, new Date());
	donaciones.add(d);
	Refugio refugio = this.getRefugio();
	refugio.setLiquidez(refugio.getLiquidez() + d.getCantidad());
	}
}
```

Clase enumerada Estado Animal

```
public enum EstadoAnimal {
    disponible, adoptado, enTratamiento
}
```

Clase Refugio

```
public class Refugio {
	private Float liquidez;
	private ArrayList<Animal> animalesRefugiados;
	private ArrayList<Animal> animalesRegistrados;

	public Refugio(Float liquidez) {
	this.liquidez = liquidez;
	this.animalesRefugiados = new ArrayList<>();
	this.animalesRegistrados = new ArrayList<>();
	}
	/**
	*
	* @param a
	*/
```

```
	public void registrar(Animal a) {
	a.setEstado(EstadoAnimal.disponible);
	this.animalesRegistrados.add(a);
	this.animalesRefugiados.add(a);
	}

	public Float getLiquidez() {
	return liquidez;
	}

	protected void setLiquidez(Float liquidez) {
	this.liquidez = liquidez;
	}

	public ArrayList<Animal> getAnimalesRefugiados() {
	return animalesRefugiados;
	}
	public ArrayList<Animal> getAnimalesRegistrados() {
	return animalesRegistrados;
	}
}
```

Clase abstracta Socio

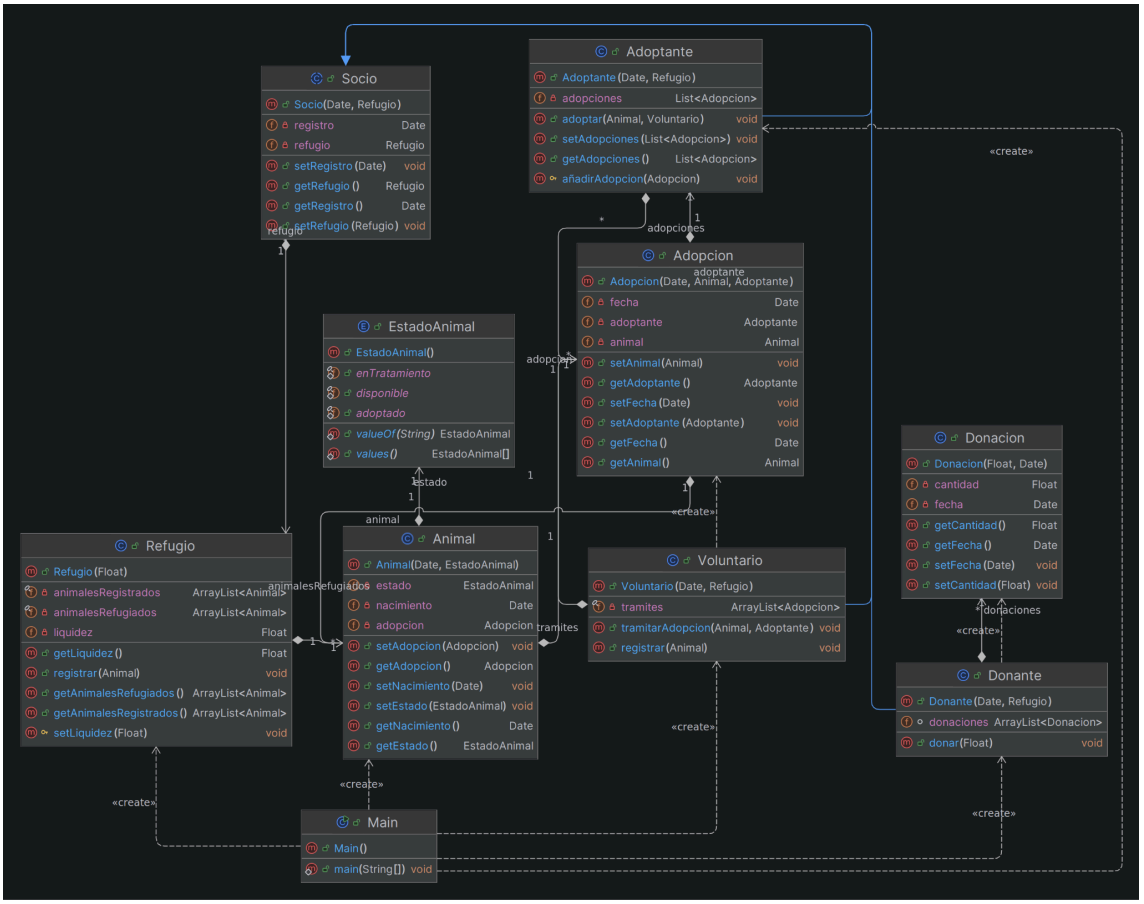
```
public abstract class Socio {
	private Date registro;
	private Refugio refugio;

	public Socio(Date registro, Refugio refugio) {
	this.setRegistro(registro);
	this.setRefugio(refugio);
	}
	public Date getRegistro() {
	return registro;
```

```
	}

	public void setRegistro(Date registro) {
	this.registro = registro;
	}

	public Refugio getRefugio() {
	return refugio;
	}

	public void setRefugio(Refugio refugio) {
	this.refugio = refugio;
	}
}
```

Clase Voluntario

```
public class Voluntario extends Socio {
	private ArrayList<Adopcion> tramites;

	public Voluntario(Date registro,Refugio refugio) {
	super(registro,refugio);
	this.tramites = new ArrayList<>();
	}

	public void tramitarAdopcion(Animal a, Adoptante ad) {
	Adopcion adopcion = new Adopcion(new Date(), a, ad);
	tramites.add(adopcion);
	a.setEstado(EstadoAnimal.adoptado);
	ad.añadirAdopcion(adopcion);

	Refugio refugio = this.getRefugio();

	if (refugio.getAnimalesRefugiados().remove(a)) {
```

```
            System.out.println("El animal ha sido adoptado y eliminado del
refugio.");
      } else {
            System.out.println("El animal no está en la lista de animales
refugiados.");
      }
      }

      public void registrar(Animal a) {
      super.getRefugio().registrar(a);
      }
}
```


Socio
Socio(Date, Refugio)
registro Date
refugio Refugio
setRegistro(Date) void
getRefugio() Refugio
getRegistro() Date
setRefugio(Refugio) void
Adoptante
Adoptante(Date, Refugio)
adopciones List<Adopcion>
adoptar(Animal, Voluntario) void
setAdopciones(List<Adopcion>) void
getAdopciones() List<Adopcion>
añadirAdopcion(Adopcion) void
Adopcion
Adopcion(Date, Animal, Adoptante)
fecha Date
adoptante Adoptante
animal Animal
setAnimal(Animal) void
getAdoptante() Adoptante
setFecha(Date) void
setAdoptante(Adoptante) void
getFecha() Date
getAnimal() Animal
EstadoAnimal
EstadoAnimal()
enTratamiento
disponible
adoptado
valueOf(String) EstadoAnimal
values() EstadoAnimal[]
Donacion
Donacion(Float, Date)
cantidad Float
fecha Date
getCantidad() Float
getFecha() Date
setFecha(Date) void
setCantidad(Float) void
Refugio
Refugio(Float)
animalesRegistrados ArrayList<Animal>
animalesRefugiados ArrayList<Animal>
liquidez Float
getLiquidez() Float
registrar(Animal) void
getAnimalesRefugiados() ArrayList<Animal>
getAnimalesRegistrados() ArrayList<Animal>
setLiquidez(Float) void
Animal
Animal(Date, EstadoAnimal)
estado EstadoAnimal
nacimiento Date
adopcion Adopcion
setAdopcion(Adopcion) void
getAdopcion() Adopcion
setNacimiento(Date) void
setEstado(EstadoAnimal) void
getNacimiento() Date
getEstado() EstadoAnimal
Voluntario
Voluntario(Date, Refugio)
tramites ArrayList<Adopcion>
tramitarAdopcion(Animal, Adoptante) void
registrar(Animal) void
Donante
Donante(Date, Refugio)
donaciones ArrayList<Donacion>
donar(Float) void
Main
Main()
main(String[]) void
«create»
adopciones
adoptante
estado
animal
tramites
donaciones
refugio
animalesRefugiados

APARTADO B.

Java es un lenguaje que solo permite herencia simple, por lo que si una instancia concreta es nuestro código es por ejemplo un Voluntario, no puede ser a su vez un Adoptante. Java no permite asignar múltiples subclases que extiendan de la misma clase padre a una misma instancia.

APARTADO C

Creamos la interfaz Rol, la clase adoptante, voluntario y donante heredan de esta.

Socio tiene una lista de roles y que también heredan.

```
public abstract class Rol{
}


public class Socio {
	private Date registro;
	private Refugio refugio;
	private List<Rol> roles = new ArrayList<>();


	public Socio(Date registro, Refugio refugio) {
	this.setRegistro(registro);
	this.setRefugio(refugio);
	}


	public void agregarRol(Rol rol) {
	if (!roles.contains(rol)) {
		roles.add(rol);
	}
	}


	public Donante getRolDonante() {
	for (Rol rol : roles) {
		if (rol instanceof Donante) {
		return (Donante) rol;
```

```
            }
        }
        return null;
        }

        public Adoptante getRolAdoptante() {
        for (Rol rol : roles) {
            if (rol instanceof Adoptante) {
            return (Adoptante) rol;
            }
        }
        return null;
        }

        public Voluntario getRolVoluntario() {
        for (Rol rol : roles) {
            if (rol instanceof Voluntario) {
            return (Voluntario) rol;
            }
        }
        return null;
        }

        public List<Rol> getRoles() {
        return roles;
        }

        public Date getRegistro() {
        return registro;
        }

        public void setRegistro(Date registro) {
```

```
        this.registro = registro;
        }

        public Refugio getRefugio() {
        return refugio;
        }

        public void setRefugio(Refugio refugio) {
        this.refugio = refugio;
        }

}
```

La otra forma que hemos pensado sería haciendo una nueva clase abstracta Rol en la que esté todo lo común a todos los roles y que Voluntario, Adoptante y Donante extiendan de rol.

Así podemos hacer que Socio no sea abstracto y contenga una lista de roles para así poder ser cualquier cosa. Este es el que elegimos así que el código más abajo es la implementación.

APARTADO D. IMPLEMENTACIÓN JAVA