Curso 2018-2019. Cuatrimestre de Otoño. 10 de Enero de 2019
Programación Orientada a Objetos (POO). Solución del examen final.

Problema 1

```
public class Simulador {
    private CalculadorRuta router;
    private Red red;
    private Map<Integer, Peticion> peticiones;
. . . }

public class Red {
    private Map<Integer, Router> nodos;
    private List<Enlace> enlaces;
. . . }

public class Peticion {
    private static int LAST_ID = 0;
    private int id;
    private int idOrigen;
    private int idDestino;
    private int volumen;
    private Ruta ruta;
. . . }

public class Router {
    private int id ;
    private Map<Integer,Enlace> enlaces ;
. . . }

OPCION 1: disponer los dos routers conectados por el enlace en un
array de dimension 2.
public class Enlace {
    private int capacidad;
    private int ocupada;
    private int longitud;
    private Router[] routers;
. . . }

OPCION 2: disponer cada uno de los routers conectados por el enlace en
un atributo diferente.

public class Enlace {
    private int capacidad;
    private int ocupada;
    private int longitud;
    private Router rOrigen;
    private Router rDestino ;
. . . }
```

Obviamente, el conjunto de métodos que hagan uso de esta información deberá ser coherente con la opción seleccionada en esta respuesta.
Las soluciones presentadas en este documento lo son para la OPCION 1

```
public class Ruta {
    private List<Enlace> enlaces;
. . . }

public abstract class CalculadorRuta {
    protected Red red ;
. . . }

public class BreadthFirstSearch extends CalculadorRuta{}

public class ConstrainedBreadthFirstSearch extends CalculadorRuta{}

public class RedReader {}
```

Problema 2

CONSTRUCTOR DE Router:

```
    public Router(int id) {
        this.id = id;
        this.enlaces = new HashMap<Integer, Enlace>() ;
    }
```

CONSTRUCTOR DE Peticion:

Para gestionar la generación de un identificador único para cada petición hay que añadir un atributo estático y privado a esta clase, inicializarlo a 0, e incrementarlo cada vez que se crea una nueva, asignando su valor al atributo id del objeto creado, tal y como se muestra a continuación:

```
    private static int LAST_ID = 0;
```

OPCION 1:

```
    public Peticion(int idOrigen, int idDestino, int volumen) {
        this.idOrigen = idOrigen;
        this.idDestino = idDestino;
        this.ces = volumen;
        Peticion.LAST_ID++;
        this.id = Peticion.LAST_ID;
    }
```

OPCION 2:

```
    public Peticion(int ces, Router rOr, Router rDest) {
        this.ces = ces;
        this.idOrigen = rOr.getId() ;
        this.idDestino = rDest.getId() ;
        Peticion.LAST_ID++;
        this.id = Peticion.LAST_ID;
```

```
}
```

Problema 3

```
    public void anotaEnlace(Enlace enlace) throws EnlaceException{
        Router rOrigen = enlace.getRouters()[0] ;
        Router rDestino = enlace.getRouters()[1] ;
        if(this!=rOrigen){
            throw new EnlaceException("El enlace no conecta este nodo
origen, de identificador "+ id + " con ningún otro nodo") ;
        }
        this.enlaces.put(rDestino.getId(), enlace) ;
    }


public Enlace getEnlaceHaciaRouter(int idRouter){
        return this.enlaces.get(idRouter) ;
    }
```

Problema 4

```
CONSTRUCTOR DE Enlace:
    public Enlace(int capacidad, int longitud, Router routOrigen,
Router routDestino) throws EnlaceException {
        this.capacidad = capacidad;
        this.longitud = longitud;
        this.ocupada = 0;
        this.routers = new Router[2] ;
        this.routers[0] = routOrigen ;
        this.routers[1] = routDestino ;
        this.routers[0].anotaEnlace(this);
    }


    public boolean conectaEstosRouters(int idRouterOrigen, int
idRouterDestino){
        if( (idRouterOrigen==this.routers[0].getId() &&
idRouterDestino==this.routers[1].getId())){
            return true ;
        }
        return false ;
    }
```

Problema 5

EN CLASE Red.

```
    public Enlace getEnlaceQueConecta(int idRouterOrigen, int
idRouterDestino) {
        for (Enlace enlace : this.enlaces) {
            if (enlace.conectaEstosRouters(idRouterOrigen,
idRouterDestino)) {
                return enlace;
            }
        }
        return null;
```

```
    }
```

Problema 6

EN CLASE RedRead

```
    protected Enlace crearEnlaceDeLinea(String linea, Red red)
            throws FormatoEnlaceException, EnlaceException {
        linea = linea.trim();
        String partes[] = linea.split(" ");
        if (partes.length != 4) {
            throw new FormatoEnlaceException("Error: no hay 4 valores
para especificar un enlace");
        }
        int idRouterOr, idRouterDest, capacidad, longitud;
        try {
            idRouterOr = Integer.parseInt(partes[0]);
            idRouterDest = Integer.parseInt(partes[1]);
            capacidad = Integer.parseInt(partes[2]);
            longitud = Integer.parseInt(partes[3]);
            return new Enlace(capacidad, longitud,
red.getRouters().get(idRouterOr), red.getRouters().get(idRouterDest));
        } catch (NumberFormatException ex) {
            throw new FormatoEnlaceException("Error al intentar leer
un entero  de la línea. Revise dicha línea.");
        }
    }

    public Red readRedFromFile(String fName) throws RedReaderException
{
        Red red = new Red();
        String linea = "";
        int numRouters = 0;
        Router router;
        Scanner reader = null;
        try {
            reader = new Scanner(new FileInputStream(fName));
        } catch (FileNotFoundException ex) {
            throw new RedReaderException(ex.getMessage());
        }
        // Primera línea: entero con número de routers
        if (reader.hasNextLine()) {
            linea = reader.nextLine();
            try {
                numRouters = Integer.parseInt(linea);
            } catch (NumberFormatException ex) {
                throw new RedReaderException(ex.getMessage());
            }
        }
        // Ahora creamos todos los routers con sus identificadores
        for (int i = 0; i < numRouters; i++) {
```

```
            router = new Router(i + 1);
            red.getRouters().put(router.getId(), router);
        }
        // Ahora leemos las siguientes líneas e intentamos crear los
enlaces
        Enlace enlace = null ;
        while (reader.hasNextLine()) {
            linea = reader.nextLine();
            linea = linea.trim();
            if (!linea.isEmpty()) {
                try {
                    enlace = this.crearEnlaceDeLinea(linea, red);
                } catch (FormatoEnlaceException ex) {
                    throw new RedReaderException(ex.getMessage());
                } catch (EnlaceException ex) {
                    throw new RedReaderException(ex.getMessage());
                }
                /*
                Tambien es posible hacer un catch(Exception ex) en
lugar de los
                dos catches, ya que el tratamiento es idéntico.
                */
                red.getEnlaces().add(enlace);
            }
        }
        return red;
    }
```

Problema 7

En **BreadthFirstSearch**:

```
    public Integer[] runAlg(int idOrigen, int volumen) {
        Integer[] mejoresConexiones = new
Integer[this.red.getRouters().size()];
        Set<Integer> visitados = new HashSet<Integer>();
        List<Integer> cola = new LinkedList<Integer>();
        cola.add(new Integer(idOrigen));
        Router routerRef = null ;
        visitados.add(idOrigen) ;
        while (!cola.isEmpty()) {
            routerRef = this.red.getRouters().get(cola.remove(0)) ;
            Collection<Enlace> enlaces =
routerRef.getEnlaces().values() ;
            int idRouter = routerRef.getId() ;
            for(Enlace enlace: enlaces){
                int routerConectado = enlace.getRouters()[1].getId() ;
                if(!visitados.contains(routerConectado)){
                    mejoresConexiones[routerConectado-1] = idRouter ;
                    visitados.add(routerConectado) ;
                    cola.add(routerConectado) ;
```

```
                }
            }
        }
        return mejoresConexiones ;
    }
}
```

En **CalculadorRuta**:

```
    public Ruta calculaRuta(int idOrigen, int idDestino, int volumen)
    throws RutaException {
        Ruta ruta = new Ruta();
        int maximoId = this.red.getRouters().size() ;
        if(idOrigen<1 || idOrigen>maximoId || idDestino < 1 ||
                idDestino>maximoId){
            throw new RutaException("ERROR: Uno o ambos
identificadores están fuera de rango.") ;
        }
        if(idOrigen==idDestino){
            throw new RutaException("ERROR: el nodo origen y el
destino son el mismo nodo.") ;
        }
        Integer[] mejoresConexiones = this.runAlg(idOrigen);
        int idRouterInspeccionado = idDestino ;
        while(idRouterInspeccionado!=idOrigen){
            Integer idRouterPrevio =
mejoresConexiones[idRouterInspeccionado-1] ;
            if(idRouterPrevio==null){
                throw new RutaException("No se puede llegar a nodo con
identificador " + idRouterInspeccionado) ;
           }
            Enlace enlace =
this.red.getEnlaceQueConecta(idRouterPrevio, idRouterInspeccionado) ;
            if(enlace==null){
                throw new RutaException("Los nodos " +
idRouterInspeccionado + " y " +
                        idRouterPrevio + " no están conectados por
ningún enlace. Revise "
                        + "la red o la implementación del método
runBreadthFirstSearchAlg") ;
          }
            ruta.getEnlaces().add(0,enlace);
            idRouterInspeccionado = idRouterPrevio ;
        }
        return ruta ;
   }
```