

Tecnología de la Programación 2 - Curso 2021/2022

Convocatoria ordinaria (11/05/2022) - Duración: 3 horas

Puntuación máxima del examen: **10 puntos**

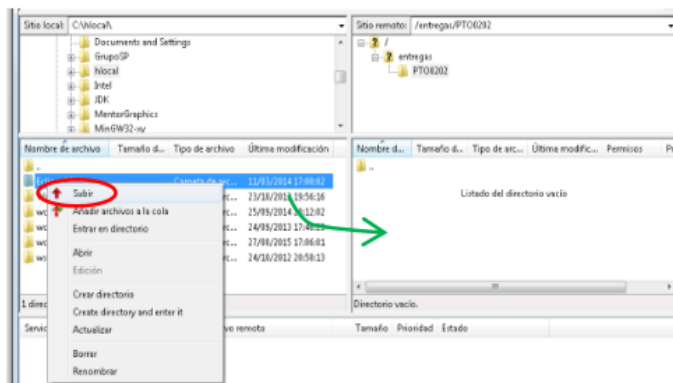
Puntuación mínima en el examen para poder aprobar la asignatura: **5 puntos**

(Grupos B,C,F, e I)

INSTRUCCIONES

1. **Descarga tu Práctica 2 del Campus Virtual**, ponla a funcionar en Eclipse y comprueba que funciona correctamente.
2. Crea un fichero de texto **cambiosExamen.txt** en la raíz de tu proyecto (dentro de **src**). En este fichero deberás incluir tu nombre completo y el nombre de todos los ficheros **.java** que modifiques. Además puedes incluir otros comentarios relevantes sobre tu solución.
3. **El código entregado debe compilar** y, no debe romper la encapsulación de las clases (acceso a atributos privados y protegidos desde clases externas, utilización de atributos públicos, etc.). **Si el código no compila o rompe encapsulación de clases, tendrá la calificación de 0 puntos en la pregunta correspondiente.**
4. En la corrección se valorará el funcionamiento, la claridad del código y el uso conveniente de los medios proporcionados por la Programación Orientada a Objetos, así como el buen uso del patrón de diseño MVC.

INSTRUCCIONES DE ENTREGA



Para entregar la solución del examen, crea un fichero **NombreApellidos.zip** (usar sólo **ZIP**, no **RAR**, ni **7z**). En él debes incluir todo el proyecto una vez limpiado de archivos intermedios (es decir, sin el directorio **bin**, que contiene los **.class**). Haz doble clic en el icono del escritorio denominado: **“EXAMENES en LABs entregas”**, y dentro de la ventana que aparece, haz doble clic en **“ALUMNOS entrega de prácticas y exámenes”**. Se abre otra ventana en la que debes seleccionar el archivo **zip**

en el panel inferior izquierdo y arrastrarlo al panel inferior derecho (o utiliza el botón derecho del ratón y la opción **Subir**), como se indica en la figura. Antes de abandonar el laboratorio debes pasar por el puesto del profesor para asegurarte de que lo que se ve en el puesto del profesor es lo que has entregado (**comprobando el tamaño del archivo**) y firmar en la hoja de entregas (usar tu bolígrafo para evitar el contacto).

Pregunta 1 [2.5 puntos]

Añade el código necesario que permite a un cruce determinado cambiar las estrategias de extracción de vehículos de su cola así como la estrategia de conmutación de luces durante la simulación. Para ello implementa los siguientes apartados:

1. Añadir los métodos `set_dqs` y `set_lss` a la clase `Junction`, para permitir modificar las estrategias correspondientes. Ambos métodos reciben una estrategia como parámetro lanzando una excepción si la estrategia introducida tiene un valor nulo.
2. Añadir el código necesario para un evento que permita cambiar una o ambas estrategias en un momento determinado.

El JSON de entrada para el evento es el siguiente:

```
{
  "type" : "set_strategies",
  "data" : {
    "time" : 343,
    "id" : "j1",
    "lss" : { "type" : "round_robin_lss", "data" : { "timeslot" : 5 } },
    "dqs" : { "type" : "move_first_dqs", "data" : {} }
  }
}
```

donde "id" es el identificador del cruce, y "lss" y "dqs" son las nuevas estrategias. Estas estrategias son opcionales, pero se debe proporcionar al menos una de ellas, de lo contrario, se generará una excepción correspondiente.

En la implementación se deberá incluir las clases del evento, el constructor (builder) utilizados para la creación de dicho evento e integrarlo en la clase `Main`. Utiliza el archivo `ex2_set_strategies.json` proporcionado para probar tu implementación.

Pregunta 2 [3.5 puntos]

Decimos que un vehículo “v” ha cruzado una localización dada “n” de la carretera “r” si:

1. Cuando comienza **advance**, el vehículo “v” está en la carretera “r” en una localización *menor que* “n”; y
2. Al final del avance, el vehículo “v” está en la carretera “r” en una localización *mayor o igual que* “n”.

Dadas una carretera “r” y una localización “n”, modifica el **modo consola** para imprimir **al final de la simulación** el número de vehículos que han cruzado esa localización, por ejemplo:

La localización 100 de la carretera r1 ha sido cruzada 4 veces

Los valores de “r” y “n” son proporcionados usando la opción command-line **-l** (p.ej., **-l r1:500**). Puedes usar el código que proporcionamos más abajo para procesar esta opción. Fíjate que si esta opción no se proporciona en command-line, no podrás contar nada.

Para implementar este ejercicio no se puede modificar ninguna clase del simulador, solo se puede modificar la clase Main y añadir nuevas clases. Tampoco se puede invocar el método run dentro de un bucle en la clase Main (mantén las llamadas a run como están).

```
// declara este atributo en la clase Main.java
private static Pair<String, Integer> _loc = null;

// añade este comando en buildOptions
cmdLineOptions.addOption(Option.builder("l").longOpt("loc").hasArg().desc("...").build());

// declara este método en Main.java y llámalo desde el método parseArgs
private static void parseLocation(CommandLine line) throws ParseException {
    String l = line.getOptionValue("l");
    if (l != null) {
        try {
            int i = l.lastIndexOf(':');
            _loc = new Pair<>(l.substring(0, i), Integer.parseInt(l.substring(i + 1)));
        } catch (Exception e) {
            throw new ParseException("Invalid location: " + l);
        }
    }
}
```

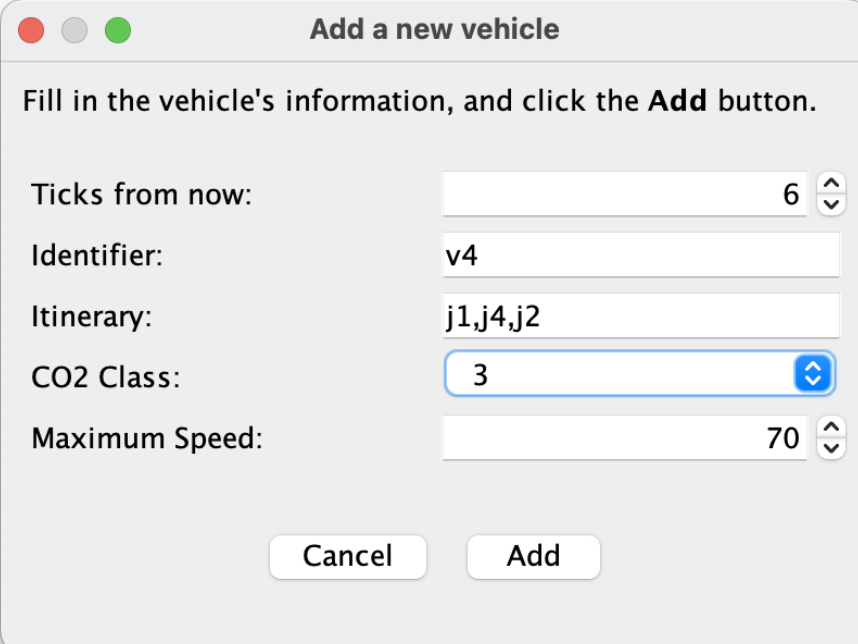
Pregunta 3 [4 puntos]

Añade un botón a la barra de herramientas de forma que al hacer clic, muestra un diálogo que permite la inserción de un nuevo vehículo a la simulación tras un número determinado de “ticks”. Para el icono del botón, puedes usar el `car_front.png`.

El identificador y el itinerario del vehículo serán proporcionados a través de componentes de tipo `JTextField`, la clase de contaminación del vehículo será proporcionado a través de un componente de tipo `ComboBox`, y la velocidad máxima del vehículo será proporcionada a través de un componente de tipo `JSpinner`. El itinerario será proporcionado por el usuario a través del componente de tipo `JTextField` mencionado anteriormente, como una cadena de identificadores de cruces, separados por coma, para la conversión de la cadena proporcionada en una lista de identificadores de cruces, puedes utilizar `Arrays.asList(str.split("\\s*,\\s*))`.

Debes de tratar todos los posibles errores mostrando un mensaje de error utilizando el método `JOptionPane.showMessageDialog`, no obstante, no tienes que comprobar si los cruces proporcionados existen o si existe otro vehículo con el mismo identificador.

Para la corrección del ejercicio, se tendrá en cuenta la funcionalidad, el aspecto estético (como de similar es, comparado con la interfaz gráfica de abajo), y el tratamiento de los posibles errores.



The image shows a Java Swing dialog box titled "Add a new vehicle". It has a standard macOS-style title bar with red, yellow, and green window control buttons. The dialog contains the following elements:

- A label: "Fill in the vehicle's information, and click the **Add** button."
- A row for "Ticks from now:" with a text field containing "6" and a `JSpinner` control.
- A row for "Identifier:" with a text field containing "v4".
- A row for "Itinerary:" with a text field containing "j1,j4,j2".
- A row for "CO2 Class:" with a `ComboBox` showing "3".
- A row for "Maximum Speed:" with a text field containing "70" and a `JSpinner` control.
- At the bottom, there are two buttons: "Cancel" and "Add".