ICB0009-UF2-PR01

Práctica entregable (Enunciado)

Guía para el alumno

El alumno debe de entregar la práctica enunciada en este documento antes del cierre programado en el calendario en la tarea habilitada en la plataforma. SALLENET. La nota de la práctica se publicará en la plataforma dentro del plazo indicado en la normativa de evaluación.

El código se ha de desarrollar en un repositorio de GitHub, sincronizándolo periódicamente, para que se vea su historia. Mientras esté en desarrollo, el repositorio será privado, cuando acabéis, lo haréis público para que yo pueda acceder.

El entregable es un documento (no necesariamente este) con el enlace al repositorio de GitHub:

- Para una mayor facilidad de corrección se deben entregar los códigos de **cada Tarea por separado**, en proyectos diferentes del mismo repo. La organización por carpetas es clave. "Ejercicio1" será una carpeta que contendrá tres subcarpetas, "Tarea1", "Tarea2" y "Tarea3. Así mismo, la carpeta "Ejercicio2" tendrá cinco subcarpetas, "Tarea1", …, "Tarea5" y la carpeta "Ejercicio3" tendrá una subcarpeta "Tarea1". Cada una de las 9 subcarpetas serán un proyecto independiente con su README.md dentro de un mismo repositorio.
- En los ficheros de código debéis añadir documentación explicando las clases y métodos introducidos.
- Cada proyecto debe tener un README.md en Mark Down, con explicaciones del propósito del código y capturas de pantalla. Para realizar la captura se utilizará la tecla "imp pant" o equivalente del teclado, y luego al final de documento y dentro de la página en blanco habilitada para cada captura, se realiza el "pegado" o "paste". Si fuera necesario se ajustará el tamaño.
- En el Mark Down también deberéis incluir respuestas a las preguntas planteadas y añadir esquemas realizados con cualquier herramienta, incluso añadir la foto de un dibujo hecho a mano. Estos esquemas pueden servir para visualizar los procesos e hilos del programa. También se pueden añadir tablas, escribir texto, insertar enlaces externos, etc.

El código contará el 50% de la nota y el README.md el otro 50%. Veréis que cada Tarea tiene asignada una puntuación.

El documento entregado tendrá el siguiente nombre y contendrá el enlace al repositorio.

ICB0009-UF2-PR01-"username".doc "username" = nombre de usuario del alumno en la plataforma Ejemplo: ICB0009-UF2-PR01-raulgarciaflores.doc

GESTIÓN ATENCIÓN HOSPITALARIA

Explicación previa

En esta práctica simularemos la gestión de la atención hospitalaria.

El hospital tiene:

• Sala de espera: Lugar donde los pacientes esperan a ser atendidos, con una capacidad de 20 personas.

• Médicos: 4 médicos que pueden atender a un paciente a la vez.

• Máquinas de diagnóstico: 2 máquinas de diagnóstico que se pueden usar con un paciente a la vez.

• Prioridades: Los pacientes se clasifican en tres niveles de prioridad:

- Emergencias (nivel 1): Atendidos primero.
- Urgencias (nivel 2): Atendidos después de las emergencias.
- Consultas generales (nivel 3): Atendidos al final.

La práctica está formada por 3 ejercicios con diversas tareas.

Cada tarea se debe entregar de tal manera que se pueda revisar por separado. Aunque la práctica es evolutiva, cada tarea debe ser escrita en código diferente, creando un proyecto nuevo.

Ejercicio #1 – Atención médica

En este ejercicio nos centraremos en preparar la atención médica a pacientes. Para ello realizaremos 3 tareas.

Ejercicio #1 – Consulta médica – Tarea #1

En esta tarea hay que simular la llegada de 4 pacientes al hospital y su atención médica en las 4 consultas médicas disponibles. Llega un paciente cada 2 segundos. Hay 4 médicos. Un médico atiende a los pacientes de forma individual. El tiempo de atención es siempre de 10 segundos. Los médicos se asignarán de forma aleatoria del 1 al 4. Si un médico ya está atendiendo a un paciente, no podrá a ninguno otro.

Cada vez que llega un paciente se mostrara un mensaje por pantalla junto al número de llegada (Paciente 1, 2, 3 o 4). Cada vez que salga un paciente de consulta se mostrara un mensaje por pantalla.

PREGUNTAS:

¿Cuántos hilos se están ejecutando en este programa? Explica tu respuesta.

[Respuesta]

¿Cuál de los pacientes entra primero en consulta? Explica tu respuesta.

[Respuesta]

¿Cuál de los pacientes sale primero de consulta? Explica tu respuesta.

[Respuesta]

1,5 PUNTOS

Ejercicio #1 – **Pacientes** con datos – Tarea #2

En esta tarea el objetivo es poder pasar datos a cada paciente de tal manera que tengan comportamientos diferentes. Cada paciente tiene asignado:

- Un identificador único, un numero entero aleatorio del 1 al 100.
- Un tiempo de llegada al hospital (en segundos, empezando en 0 cuando llega el primer paciente).
- El tiempo en consulta con un médico que necesita (un numero aleatorio entre 5 y 15 segundos).
- El estado:
    - Espera: ha llegado al hospital per aún no ha entrado en consulta
    - Consulta: ha entrado en consulta
    - Finalizado: ha finalizado la consulta

La clase que se debe programar para contener esta información debe ser:

```
public class Paciente
{
    public int Id {get; set;}
    public int LlegadaHospital {get; set;}
    public int TiempoConsulta {get; set;}
    public int Estado {get; set;}


    public Paciente (int Id, int LlegadaHospital, int TiempoConsulta)
    {
        this.Id = Id;
        this.LlegadaHospital = LlegadaHospital;
        this.TiempoConsulta = TiempoConsulta;
    }
}
```

Ahora los pacientes estarán en consulta el tiempo indicado en TiempoConsulta. También debemos mostrar por pantalla el Id del paciente, la prioridad y el número de llegada (si ha llegado el 1, 2, 3 o 4).

¿Cuál de los pacientes sale primero de consulta? Explica tu respuesta.

[Respuesta]

1 PUNTO

Ejercicio #1 – **Visualización del avance**– Tarea #3

Debéis mostrar por consola la información del paciente, junto con sus cambios de estado, el orden de llegada (1 a 4) y el tiempo entre cambios de estado. En cada cambio de estado debe visualizarse el cambio.

Paciente <Id>. Llegado el N. Estado: <estados>. Duración: <S> segundos.

Ejemplo de visualización:

Paciente 12. Llegado el 1. Estado: Finalizado. Duración Consulta: 10 segundos.
Paciente 34. Llegado el 2. Estado: Consulta. Duración Espera: 0 segundos.
Paciente 53. Llegado el 3. Estado: Consulta. Duración Espera: 0 segundos.
Paciente 21. Llegado el 4. Estado: Consulta. Duración Espera: 0 segundos.
Paciente 12. Llegado el 1. Estado: Consulta. Duración Espera: 0 segundos.

Pregunta:

¿Has decidido visualizar información adicional a la planteada en el ejercicio? ¿Por qué? Plantea qué otra información podría ser útil visualizar.

[Respuesta]

1 PUNTO

Ejercicio #2 – Unidades de diagnóstico – Tarea #1

Vamos a suponer que algunos pacientes requieren realizar pruebas en equipos limitados (por ejemplo, un escáner corporal que puede escanear el cuerpo o alguna parte del cuerpo). Vamos a suponer que el hospital tiene 2 máquinas y como mucho un paciente puede usarla al mismo tiempo.

- Ampliad la clase Paciente con el atributo requiereDiagnostico (booleano) para indicar si necesita diagnóstico adicional. Se tiene que generar aleatoriamente.
- Los pacientes entran primero en consulta y a continuación se hacen las pruebas si es necesario.
- Simular pruebas de diagnóstico con un tiempo adicional (15 segundos).
- Ampliad la clase Estado permitiendo los siguientes estados:
    - EsperaConsulta: ha llegado al hospital pero aún no ha entrado en consulta
    - Consulta: ha entrado en consulta, pero aún no ha salido
    - EsperaDiagnostico: ha acabado la consulta, requiere diagnóstico, pero aún no ha sido expuesto a la máquina de diagnostico
    - Finalizado: ha finalizado la consulta y el diagnóstico si es que era necesario
- Actualizar la visualización del avance.

Pregunta:

¿Los pacientes que deben esperar para hacerse las pruebas diagnostico entran luego a hacerse las pruebas por orden de llegada? Explica que tipo de pruebas has realizado para comprobar este comportamiento.

[Respuesta]

1 PUNTO

Ejercicio #2 – Unidades de diagnóstico – Tarea #2

Ahora vamos a crear una sincronización para realizar las pruebas. En la tarea anterior pasaba el primer paciente que llegaba, ahora deberán pasar por orden, primero pasará el paciente 1 (el primero en llegar al hospital), luego el 2 (el segundo en llegar), luego el 3 y finalmente el 4. Este orden se deberá mantener independientemente del orden de salida de la consulta.

Por ejemplo, si el paciente 1 es el que más tiempo lleva en consulta, todos los demás deberán esperar a que el 1 realice las pruebas antes que ellos.

Pregunta:

Explica la solución planteada en tu código y porqué las has escogido.

Parte 2:

Plantea otra posibilidad de solución a la que has programado.

[Respuesta]

1,5 PUNTO

Ejercicio #2 – Más pacientes – Tarea #3

Mantendremos las 4 consultas médicas con los 4 médicos y las dos máquinas de diagnóstico, pero añadiremos más pacientes. En total 20 pacientes. Llegaran de forma secuencial, uno cada 2 segundos. Tendrán que esperar, estado EsperaConsulta, si no hay ninguna consulta médica disponible.

Pregunta:

Explica el planteamiento de tu código y plantea otra posibilidad de solución a la que has programado y porqué has escogido la tuya.

[Respuesta]

Pregunta:

¿Los pacientes que deben esperar entran luego a la consulta por orden de llegada? Explica que tipo de pruebas has realizado para comprobar este comportamiento.

[Respuesta]

1,5 PUNTOS

Ejercicio #2 – Prioridades de los pacientes – Tarea #4

Vamos a suponer que los pacientes al llegar al hospital se les asigna un nivel de prioridad, de entre en tres:

- Emergencias (nivel 1): Atendidos primero.
- Urgencias (nivel 2): Atendidos después de las emergencias.
- Consultas generales (nivel 3): Atendidos al final.

Vamos a realizar lo siguiente:

- Ampliad la clase Paciente con el atributo prioridad (int) para indicar la prioridad del paciente
- Los pacientes en espera entraran por orden de prioridad. Si hay varios con máxima prioridad entraran según el orden de llegada.

Pregunta:

Explica el planteamiento de tu código y plantea otra posibilidad de solución a la que has programado y porqué has escogido la tuya.

[Respuesta]

1,5 PUNTO

Ejercicio #2 – Estadísticas y logs – Tarea #5

Ahora, al final de la simulación mostrar unas estadísticas:

- Número total de pacientes atendidos por prioridad.
- Tiempo promedio de espera por paciente.
- Uso promedio de las máquinas de diagnóstico.

--- FIN DEL DÍA ---

Pacientes atendidos:

- Emergencias: 5
- Urgencias: 10
- Consultas generales: 8

Tiempo promedio de espera:

- Emergencias: 2s
- Urgencias: 4s
- Consultas generales: 6s

Uso promedio de máquinas de diagnóstico: 75%

Pregunta:

¿Puedes explicar tu código y porque has decidido hacerlo así?

[Respuesta]

0,5 PUNTO

Ejercicio #3 – Pacientes infinitos – Tarea #1

Parte 1:

En este ejercicio modificaremos la generación de pacientes, de tal manera que haya un “generador” de pacientes variable.

Este generador de pacientes, cada 2 segundos va a generar un nuevo paciente con datos diferentes de tiempo de consulta, prioridad y si requiere diagnostico con máquina o no.

Parte 2:

Una vez creado el generador comprobarás para diferentes cantidades de pacientes (N = 50, 100, 1000) si las diferentes tareas realizadas en el ejercicio 2 funcionan o no, qué comportamientos no previstos detectas y como adaptarías tu solución ante este nuevo escenario.

Tarea 1, ¿cumple requisitos?

[Pruebas]

[Explicación]

Tarea 2, ¿qué comportamientos no previstos detectas?

[Pruebas]

[Explicación]

Tarea 3, ¿Cómo adaptarías tu solución?

[Explicación]

0,5 PUNTO