PSP - U3 Actividades (productor-consumidor)

Descargar estos apuntes

Índice

Warning

Para todas las aplicaciones, **las excepciones deben controlarse en el código del programa**. Si no se indica lo contrario, en el bloque catch siempre se mostrará la pila de llamadas usando el método `printStackTrace()` de la excepción capturada junto con la versión traducida del mensaje de error asociado, método `getLocalizedMessage()`.

Aspectos a tener en cuenta

En estas actividades volvemos a usar la nomenclatura de clases y proyectos explicada en la hoja de actividades de autoevaluación de la U2

Ponerle nombre a los hilos

Es importante darle nombre a los threads y mostrar ese nombre en las salidas por consola. De esta forma se puede identificar qué hilo es el que está produciendo esa información.

Uso de colores en la consola

Para diferenciar las salidas de los diferentes hilos, siempre que sea posible, podéis consultar el (Anexo I de la Unidad 2).

Tiempos de simulación

Realiza las simulaciones de las actividades usando tiempos proporcionales a los que se indican en el enunciado.

- Por ejemplo, si habla de 1h y 5h, se pueden usar 10ms/50ms o bien 2ms/10ms, 100ms/500ms, 1s/5s.

Parametrización de las actividades

Siempre que sea posible, parametriza las actividades para que se puedan ejecutar con diferentes valores.

La parametrización se puede hacer mediante argumentos de la línea de comandos, ficheros de configuración, etc.

Lo ideal es que en la clase principal se definan/lean constantes y que el resto de clases reciban los valores que necesiten a través de sus `constructores`.

De esta forma podemos modificar fácilmente los valores de configuración sin tener que modificar el código de las clases.

30. Clase psp.actividades.U3A30_Restaurante

Considera un restaurante que tiene un cocinero y un camarero. El camarero debe esperar a que el cocinero prepare una comida.Cuando el cocinero tiene una comida preparada, se lo notifica al camarero (toca una campana), para que sirva la comida y vuelva a esperar.

El programa que lo modela mostrará cuando el cocinero ha preparado un plato de comida y cuando el camarero lo sirve.
El camarero deberá servir los mismos platos que ha cocinado el cocinero, y en el mismo orden en que éste los ha preparado.

Realiza la simulación con las siguientes clases:

- **U3A30_Carta**: dispondrá de un método llamado “elegirComida” que devuelva aleatoriamente el nombre de un plato de comida de entre los que ofrezca el restaurante (la carta del restaurante).
- **U3A30_Plato**: Esta clase proporcionará los métodos “prepara” y “sirve” para depositar la comida en él y servirla respectivamente.
- **U3A30_Camarero**: El camarero se ejecutará el mismo número de iteraciones que el cocinero. En cada uno de ellas toma la comida del plato y la sirve. Entre cada iteración duerme un tiempo aleatorio entre 1 y 3 minutos.
- **U3A30_Cocinero**: El cocinero preparará 8 platos. Cada vez dejará en el plato de comida una comida de la carta elegida al azar. Entre cocinado y cocinado se quedará descansando durante un tiempo aleatorio entre 2 y 4 minutos.
- **U3A30_Restaurante**: pondrá en marcha la aplicación.

> **Cambio los parámetros del problema**
>
> Si modificas los tiempos de espera del camarero y/o del cocinero, puedes comprobar la simulación bajo distintos escenarios, obteniendo resultados diferentes.
>
> Puedes forzar a que el camarero siempre espere al cocinero, o que el cocinero siempre tenga la comida lista cuando el camarero vaya a servirla.
>
> Puedes llegar incluso a eliminar el tiempo de espera de uno de los dos, o de los dos, para ver cómo se comporta el programa en esos casos.

A continuación se muestra un ejemplo de ejecución del programa:

```
Cocinero ha preparado un nuevo plato de (Lentejas)
Camarero ha servido un plato de (Lentejas)
Camarero esperando que se prepare un nuevo plato
Cocinero ha preparado un nuevo plato de (Lentejas)
Camarero intentando servir (Lentejas)
Camarero ha servido un plato de (Lentejas)
Cocinero ha preparado un nuevo plato de (Ensalada mixta)
Camarero ha servido un plato de (Ensalada mixta)
Camarero esperando que se prepare un nuevo plato
Cocinero ha preparado un nuevo plato de (Sopa de verduras)
Camarero intentando servir (Sopa de verduras)
Camarero ha servido un plato de (Sopa de verduras)
Cocinero ha preparado un nuevo plato de (Ternera plancha)
Camarero ha servido un plato de (Ternera plancha)
Camarero esperando que se prepare un nuevo plato
Cocinero ha preparado un nuevo plato de (Macarrones con tomate)
Camarero intentando servir (Macarrones con tomate)
Camarero ha servido un plato de (Macarrones con tomate)
Camarero esperando que se prepare un nuevo plato
Cocinero ha preparado un nuevo plato de (Ternera plancha)
Camarero intentando servir (Ternera plancha)
Camarero ha servido un plato de (Ternera plancha)
Camarero esperando que se prepare un nuevo plato
Cocinero ha preparado un nuevo plato de (Arroz de marisco)
Camarero intentando servir (Arroz de marisco)
Camarero ha servido un plato de (Arroz de marisco)
```

31. Clase psp.actividades.U3A31_Colecta

Tenemos a 4 personas (colectores) haciendo una colecta.

Según vaya el día, cada persona tardan entre 10 y 200 minutos en conseguir una cantidad de entre 4€ y 25€.
La colecta termina cuando se llega a recoger 2000€.

Realiza varias simulaciones para ver cuánto tiempo tardan en completar la colecta y muestra información de lo que ha recogido cada una de las 4 personas.

32. Clase psp.actividades.U3A32_Colecta2

> Empieza a partir de la solución del problema anterior.

Ahora, además de colectores, hay encargados de recaudar el dinero. Estos se encargan de recoger el dinero de los colectores y llevarlo a la oficina.

Los recaudadores, en un intervalo de entre 20 y 300 minutos pueden recoger entre 10€ y 40€.

Los colectores no pueden recoger más dinero cuando se ha llegado al tope, pero ahora no terminan la colecta, sólo se toman un descanso hasta que alguno de los encargados retire dinero.

Lo mismo pasa con los encargados, que si no pueden recoger dinero esperarán a que los colectores aporten más a la colecta.

El número de encargados será variable (fijadlo como una constante del programa) y se escogerá para probar toda la casuística del problema.

- Cuantos más encargados haya éstos tendrán que esperar a que los colectores aporten dinero.
- Si hay muchos más colectores que encargados, los colectores tendrán que esperar a que los encargados recojan el dinero.

El programa no tiene condición de finalización es una simulación infinita.
Se deben mostrar mensajes informando claramente de lo que está sucediendo en cada momento, y de quién está realizando cada acción.

33. Clase psp.actividades.U3A33_Casino

Queremos simular los posibles beneficios de diversas estrategias de juego en un casino. La **ruleta francesa** es un juego en el que hay una ruleta con 37 números (del 0 al 36).

- Cada 60 segundos el croupier saca un número al azar y los diversos jugadores apuestan para ver si ganan.
- Todos los jugadores empiezan con 1.000 euros y la banca (que controla la ruleta) con 50.000.
- Cuando los jugadores pierden dinero, la banca incrementa su saldo en la apuesta del jugador.

Apostar a un número

Vamos a simular que 4 jugadores apuestan a un número al azar.

- Cada uno de ellos elige un número al azar del 1 al 36 (no el 0).
- Siempre apuestan 10€ de su saldo a ese ese número.
- Si sale su número su saldo se incrementa en 360 euros (36 veces lo apostado).

Apostar a par/impar

Vamos a simular que 4 jugadores apuestan a par/impar.

- Cada uno de ellos apuestan al azar a que saldrá un número par o un número impar.
- Siempre apuestan 10€ y si ganan incrementan su saldo en 20 euros (el doble de lo apostado).

Martingala

Vamos a simular que 4 jugadores apuestan a un número al azar.

- Cada uno de ellos elige un número al azar del 1 al 36 (no el 0).
- Empiezan apostando 10€ de su saldo a ese número.
- Si ganan incrementan su saldo en 360 euros (36 veces lo apostado).
- Si pierden jugarán el doble de su apuesta anterior (es decir, 20, luego 40, luego 80, y así sucesivamente).
- Si ganan siempre multiplican su apuesta por 36.

> La banca acepta todas las apuestas pero nunca paga más dinero del que tiene.
> Si sale el 0, todo el mundo pierde y la banca se queda con el dinero de todas las apuestas.

El juego acaba cuando todos los jugadores se hayan quedado `sin blanca` o se haya producido `bancarota`.

Flujo del juego

El flujo del juego será el siguiente:

- Se muestra el saldo de cada jugador y de la banca.
- Cada jugador realiza su apuesta y espera a que se saque el número. Se muestra la apuesta realizada.
- El croupier saca un número y avisa a los jugadores.
- Se actualiza el saldo de cada jugador y de la banca.
- Se comprueba si hay algún jugador que se ha quedado sin dinero. **Éste se elimina del juego**.
- Se comprueba si la banca se ha quedado sin dinero o si no quedan jugadores. **Fin del juego**.
- Si no se ha acabado el juego, **se vuelve a empezar**.

34. Clase psp.actividades.U3A34_GrandesAlmacenes

En unos grandes almacenes hay 300 clientes agolpados en la puerta para intentar conseguir un producto del cual solo hay 100 unidades.

Por la puerta solo cabe una persona, pero la paciencia de los clientes es limitada por lo que solo harán un máximo de 10 intentos para entrar por la puerta. Si después de 10 intentos la puerta no se ha encontrado libre ni una sola vez, el cliente desiste y se marcha.

Cuando se consigue entrar por la puerta el cliente puede encontrarse con dos situaciones:

- Quedan productos: el cliente cogerá uno y se marchará.
- No quedan productos: el cliente simplemente se marchará.

Realizar la simulación en Java de dicha situación.

> **Planteamiento del problema**
>
> Tened en cuenta que dentro de este problema realmente hay dos, por un lado está el acceso a la tienda y por otro la compra del producto.

35. Clase psp.actividades.U3A35_DescansoLaboral

Tenemos a 4 personas trabajando entre 2 y 4 horas, después se preparan un bocadillo y vuelven a trabajar, y así continuamente.

La empresa tiene una pequeña despensa con productos para preparar bocadillos. Inicialmente se tienen 6 hojas de lechuga, 3 tomates y 500g de jamón.

Para cada bocadillo se necesitan 2 hojas de lechuga, 1 tomate y 100g de jamón.

Existe un servicio de catering que cada 8 horas repone la lechuga, cada 5 horas repone el tomate y cada 12 horas repone el jamón. Todos los productos, por problemas de espacio, se reponen hasta alcanzar las cantidades iniciales.

La empresa quiere realizar una simulación para ver si con este sistema los trabajadores pierden mucho tiempo esperando para preparar sus bocadillos.

> **Proveedores**
>
> Los proveedores de los productos no se quedan esperando a que los trabajadores recojan los productos, simplemente completan el stock hasta llenar la despensa.
>
> Un ejemplo de la simulación, para la reposición, sería el siguiente:
> A las 5 horas de empezar la simulación, se reponen los tomates, 3 horas después se reponen las lechugas, 2 horas después se reponen de nuevo los tomates, 2 horas más tarde se repone jamón, 3 horas después se vuelven a reponer tomates, una 1 hora después se vuelven a reponer lechugas, etc.
> Cada producto lleva su tiempo de reposición independiente según los tiempos indicados.
>
> ```
> Tomates:  ----X----X----X----X----X----X----X----X----X----X----X----X----X----X----X----X
> Lechugas: -------X-------X-------X-------X-------X-------X-------X-------X-------X-------X---
> Jamón:    -----------X-----------X-----------X-----------X-----------X-----------X-----------X---
> ```

36. Clase psp.actividades.U3A36_Parking

Se tiene un aparcamiento con N(10) plazas, numeradas de 1 a N(10), al que intentan acceder coches continuamente.

Los coches llegan con una frecuencia entre 5 y 15 minutos.

Cada coche se identifica con una matrícula (aleatoria en formato LLL 1234 -3 letras y 4 dígitos-) que queda asociada a la plaza de aparcamiento libre que se le asigna, si la hay.

- Si el parking está lleno, los coches esperan en la entrada a que haya plaza.
- Si un coche consigue aparcar, el tiempo de permanencia de los coches en el parking está entre 30 y 60 minutos.

Realiza varias simulaciones para ver si el parking se va llenando, y en cuanto tiempo lo hace, o si por el contrario nunca llega a llenarse.

Muestra información detallada de lo que va pasando, informando de cuántas plazas están ocupadas y cuántas quedan libres en cada momento. Informa también si un coche se queda esperando en la entrada.

En ete ejemplo se ha optado por asignar las plazas de aparcamiento de forma secuencial, es decir, si hay plazas libres, se asigna la primera que esté libre. Se puede optar por asignarlas de forma cíclica, aleatoria. Tienes libertad para hacerlo como más te convenga.

```
ENTRADA: Coche 1234-ABC entra en la plaza 1
    Plazas libres: 9
    Parking: [1234-ABC] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
ENTRADA: Coche 2345-DEF entra en la plaza 2
    Plazas libres: 8
    Parking: [1234-ABC] [2345-DEF] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
ENTRADA: Coche 3456-GHI entra en la plaza 3
    Plazas libres: 7
    Parking: [1234-ABC] [2345-DEF] [3456-GHI] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
SALIDA: Coche 1234-ABC abandona la plaza 1
    Plazas libres: 8
    Parking: [ ] [2345-DEF] [3456-GHI] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
ENTRADA: Coche 4567-JKL entra en la plaza 1
    Plazas libres: 7
    Parking: [4567-JKL] [2345-DEF] [3456-GHI] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
SALIDA: Coche 3456-GHI abandona la plaza 3
    Plazas libres: 8
    Parking: [4567-JKL] [2345-DEF] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

Cola de espera

Si un coche se queda esperando en la entrada, se debe mostrar un mensaje indicándolo. Puedes usar System.err para diferenciar el mensaje de COLA de los de ENTRADA y SALIDA.

```
COLA: Coche 5678-MNO espera en la entrada
```

La reentrada de un coche que ha estado esperando en la entrada se puede gestionar de dos formas:

- Se coge cualquiera de los coches que está esperando y se le asigna la plaza que ha dejado libre el coche que ha salido.
- Se coge el primer coche que se quedó esperando y se le asigna la plaza que ha dejado libre el coche que ha salido.

La segunda opción es la que ocurre en realidad, ya que los coches están en cola a la entrada del parking esperando a que quede una plaza libre. La segunda es más sencilla de implementar.

Puedes elegir cualquiera de las dos opciones, o empezar por la más sencilla y después intentar la otra.

37. Clase psp.actividades.U3A37_ParkingCamiones

Partiendo del problema anterior, ahora en el parking también pueden entrar camiones. El problema es que un camión necesita dos plazas contiguas, una par y otra impar, siendo la impar la menor de ellas (en nuestro parking, entre una plaza par y la siguiente impar hay una columna).

El número de camiones que accede a nuestro parking es mucho menor que el de coches. Escoger el número de vehículos de cada tipo teniendo en cuenta que deberían guardar una relación de 10/3.

38. Clase psp.actividades.U3A38_Puente

Necesitamos simular un sistema que controla el paso de personas por un puente, siempre en la misma dirección, para que se cumplan las siguientes restricciones.

- No pueden pasar más de tres personas a la vez,
- No puede haber más de 200kg de peso en ningún momento.

El tiempo de llegada entre dos personas está entre 1 y 30 minutos y para atravesar el puente se tarda también un tiempo entre 10 y 50 minutos. Las personas tienen un peso entre 40 y 120kg.

Realizar la simulación usando una escala de tiempo proporcional.

Muestra cuántas personas hay en el puente en cada momento, cuánto peso está soportando el puente y cuánto tiempo les queda a cada uno para terminar de cruzar el puente.

> **Información**
>
> La información que se muestra en cada momento se refiere a cada vez que se muestra información por consola.
> En cada salida se debe mostrar toda la información indicada para poder comprobar que el programa funciona correctamente.

39. Clase psp.actividades.U3A39_PuenteDobleSentido

Partimos del problema anterior. Ahora se permite el paso de personas en los dos sentidos.

Se han modificado las restricciones:

- Como máximo puede haber cuatro personas en el puente
- Pueden pasar un máximo de tres en un mismo sentido.
- El peso máximo se ha aumentado hasta los 250kg.

Amplia la información con la nueva información, indicando en qué sentido va cada persona.