

Agenda

- Colas
 - Qué es una cola
 - Métodos de colas
- Pilas
 - Qué es una pila
 - Métodos de pilas
- BP7
 - Introducción
 - Diseño
 - Sugerencias

8/4/2002

1

Qué es una pila

- Una pila es un tipo de contenedor de objetos simples
- Los objetos se insertan y se extraen según la siguiente secuencia: el último en llegar es el primero en salir (LIFO)
 - Piense en una pila como si fuese un bote de pelotas de tenis: la última insertada es la que se saca primero
- Métodos:

```
public interface Stack {  
    public boolean isEmpty();  
    public void push(Object o);  
    public Object pop() throws EmptyStackException;  
    public void clear();  
}
```

8/4/2002

2

Métodos de las pilas

- `void push(Object item)`
 - Empuja un elemento dentro de la pila
 - Tenga en cuenta que un elemento puede ser cualquier tipo de objeto (cadena, clase definida por el usuario, etc.)
- `Object pop()`
 - Extrae el elemento superior de la pila y lo devuelve al *llamador*
 - Tenga en cuenta que lo que se devuelve es un objeto genérico
 - *¿Cómo lo convertimos en el tipo de datos original?*
- `void clear()`
 - Vacía la pila
- `boolean isEmpty()`
 - Devuelve `true` si la pila está vacía y `false` si no lo está

8/4/2002

3

Qué es una cola

- Una cola es otro tipo de contenedor de objetos simples
- Los objetos se insertan y se extraen de la cola según la siguiente secuencia: el primero en llegar es el primero en salir (FIFO)
 - Piense en una cola como si fuese la cola de espera de una cabina de peaje
- **Métodos**

```
public interface Queue
{
    public boolean isEmpty();
    public void add(Object o);
    public Object remove() throws NoSuchElementException;
    public void clear();
}
```

8/4/2002

4

Métodos de las colas

- `void add(Object item)`
 - Inserta un elemento en la cola
 - Tenga en cuenta que un elemento puede ser cualquier tipo de objeto (cadena, clase definida por el usuario, etc.)
- `Object remove()`
 - Extrae el primer elemento de la cola y lo devuelve al *llamador*
 - Si la cola está vacía, arroja una excepción `NoSuchElementException`
 - Tenga en cuenta que lo que se devuelve es un objeto genérico
 - *¿Cómo lo convertimos en el tipo de datos original?*
- `void clear()`
 - Vacía la cola
- `boolean isEmpty()`
 - Devuelve `true` si la cola está vacía y `false` si no lo está

8/4/2002

5

BP7: Introducción

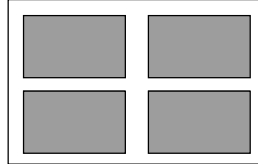
- Modela un sistema de gestión de materiales para una empresa que fabrica equipos de música
- 6 tipos de productos, cada uno con
 - un tipo de producto (0-5)
 - un número de serie (único para cada equipo, independientemente del tipo)
- 2 líneas de producción
 - FIFO (cola)
 - Línea 0: produce 8 equipos por minuto
 - Línea 1: produce 4 equipos por minuto
- 1 robot
 - Mueve los equipos desde las líneas de producción hasta el área de apilado (lo hace de 4 en 4)
 - Consulte las máquinas de helados del MIT como ejemplo (bldg 16/26, etc.)

8/4/2002

6

BP7: Introducción

- 6 áreas de apilado
 - Una para cada tipo de equipo
 - 4 pilas por área
- Pila
 - LIFO (pila)
- 1 vehículo de transporte
 - Basado en el pedido del cliente para recoger los productos de cada área de apilado
 - Retira los productos de las pilas



8/4/2002

7

BP7: Diseño

- 2 interfaces y 2 clases de implementación
 - Stack / ArrayStack
 - Para modelar las pilas de las áreas de apilado
 - Queue / ArrayQueue
 - Para modelar las líneas de producción
- 4 clases de aplicación
 - Product
 - Robot
 - StackingArea
 - TransferCar
- Clase main
 - Crea una instancia de cada componente del sistema
 - Recibe el pedido del cliente
 - Recoge y embala el pedido

8/4/2002

8

BP7 - Sugerencia: producto y línea de producto

- clase Product
 - 2 atributos: productType y serialNum
 - **ProductType (tipo de equipo) tiene valores entre 0 y 5**
 - **serialNum comienza con 0 y aumenta en 1 por cada equipo creado**
 - Constructor
 - *Obtenedores*
- ProductLine (*creada dentro del método main*)
 - Cada prodLine es un ArrayQueue

```
prodLine0 = new ArrayQueue();
prodLine1 = new ArrayQueue();
```
 - Simula 5 ciclos de producción para cada prodLine. En cada ciclo,
 - **Pregunta al usuario qué producto crear en la línea 0**
 - **Crea 8 productos de este tipo y los agrega a prodLine0**
 - **Pregunta al usuario qué tipo de producto crear en la línea 1**
 - **Crea 4 productos de este tipo y los agrega a prodLine1**

8/4/2002

9

BP7 - Sugerencia: pila y área de apilado

- Pila
 - Cada pila es un ArrayStack

```
stack0 = new ArrayStack();
```
- clase StackingArea
 - Atributos: cada StackingArea contiene 4 pilas
 - Métodos:
 - ```
public void addToStack(Product p)
{

 stackXX.push(p); //¿Cómo saber a qué pila se agrega?
}

```
    - ```
public Product getFromStack()
{
    .....
    stackYY.pop(); //¿Cómo saber de qué pila se obtiene?
}

```
 - **Para conservar un equilibrio en la altura de las pilas, rote el número de pilas a las que se agrega y de las que se obtiene (por ejemplo, comenzando por stack0, stack1, ... stack3)**
 - Sugerencia: utilice un índice para la siguiente pila

8/4/2002

10

BP7 - Sugerencia: robot y vehículo de transporte

- clase Robot
 - Recoge productos de la línea de producción y los coloca en el área de Apilado de 4 en 4.

```
void moveProduct(ArrayQueue line, StackArea s) {
    for (int i=0; i < 4; i++) { // Mueve 4 equipos cada vez
        //Extrae un producto de la línea
        //Agrega el producto a s de StackArea
    }
}
```

- clase TransferCar
 - Recoge productos según el pedido del cliente y los extrae del área de apilado correspondiente

```
public void pickProduct(StackArea[] s, int[] order) {
    for (int i=0; i < 6; i++)
        for (int j=0; j < order[i]; j++) {
            // Obtiene el producto p de StackArea
        }
}
```

8/4/2002

11

BP7 - Sugerencia: clase main

- Crea 1 robot `r`
 - Crea 1 vehículo de transporte `t`
 - Crea un *array* con 6 áreas de apilado `s`
- ```
StackArea[] s= new StackArea[6];
for (int i= 0; i < 6; i++)
 s[i]= new StackArea();
```
- Crea 2 líneas de producción: `prodLine0` y `prodLine1`
    - Simula 5 ciclos de producción para cada línea
  - El robot recoge productos de la línea 0,1,0; coloca equipos en las áreas de apilado correspondientes
    - ProdLine0 es el doble de rápida que ProdLine1

```
r.moveProduct(prodLine[0], s[product[0]]);
r.moveProduct(prodLine[1], s[product[1]]);
r.moveProduct(prodLine[0], s[product[0]]);
```

8/4/2002

12

## BP7 - Sugerencia: clase main

- Pide al usuario que facilite cantidades de pedidos de cada tipo de producto

```
int[] orderQty= new int[6];
for (int i=0; i < 6; i++) {
 String text= JOptionPane.showInputDialog("Introduzca
la cantidad de productos " + i + " para el envío:");
 orderQty[i]= Integer.parseInt(text);
}
```

- El vehículo de transporte recoge los pedidos de las áreas de apilado

```
t.pickProduct(s, orderQty);
```