

Agenda

- Queue
 - What is a queue
 - Queue methods
- Stack
 - What is a stack
 - Stack methods
- PS7
 - Overview
 - Design
 - Hints

4/8/2002

1

What is a Stack

- A stack is a type of simple object container
- Objects are inserted and removed in the last-in first-out (LIFO) fashion
 - Think of a stack as a can of tennis balls - the last ball inserted in the can is the first one taken out

- **Methods:**

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

4/8/2002

2

Stack Methods

- `void push(Object item)`
 - Pushes an item into the stack
 - Note that the item can be any object type (String, user-defined class, etc.)
- `Object pop()`
 - Pops the top item of the stack and returns it to the caller
 - Note that what returned is a generic Object
 - *How do we convert it into the original data type?*
- `void clear()`
 - Empty the stack
- `boolean isEmpty()`
 - Return true if the stack is empty and false if otherwise

4/8/2002

3

What is a Queue

- A queue is another simple type of object container
- Objects are inserted in and removed from the queue in the first-in first-out (FIFO) fashion
 - Think of a queue as a waiting line in the toll booth
- **Methods**

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

4/8/2002

4

Queue Methods

- `void add(Object item)`
 - Insert an item into the queue
 - Note that the item can be any object type (String, user-defined class, etc.)
- `Object remove()`
 - Remove the first item from the queue and return it to the caller
 - If the queue is empty, throw `NoSuchElementException`
 - Note that what is returned is a generic `Object`
 - *How do we convert it into the original data type?*
- `void clear()`
 - Empty the queue
- `boolean isEmpty()`
 - Return `true` if the queue is empty and `false` if otherwise

4/8/2002

5

PS7: Overview

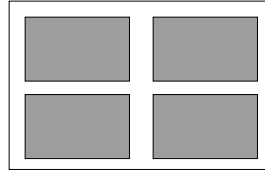
- Model a materials management system for a stereo manufacturing facility
- 6 types of product, each with a
 - product type (0 - 5)
 - serial number (unique across all stereos, regardless of type)
- 2 production lines
 - FIFO (queue)
 - Line 0: produce 8 stereos/min
 - Line 1: produce 4 stereos/min
- 1 robot
 - Move stereos from production lines to stacking area, 4 at a time
 - See the ice cream machines at MIT for an example (bldg 16/26, etc.)

4/8/2002

6

PS7: Overview

- 6 stacking areas
 - One for each stereo type
 - 4 stacks per area
- Stack
 - LIFO (stack)
- 1 Transfer Car
 - Based on customer order to pick products from corresponding stacking areas
 - Remove products from stacks



4/8/2002

7

PS7: Design

- 2 interfaces and 2 implementing classes
 - Stack / ArrayStack
 - **To model stacks within stacking areas**
 - Queue / ArrayQueue
 - **To model production lines**
- 4 application classes
 - Product
 - Robot
 - StackingArea
 - TransferCar
- Main class
 - Instantiate each component of the system
 - Take customer order
 - Pick/pack the order

4/8/2002

8

PS7 Hint: Product and Production Line

- `class Product`
 - 2 attributes: `productType` and `serialNum`
 - **ProductType (type of stereo) ranges from 0 to 5**
 - **serialNum starts with 0 and increases by 1 for each stereo built**
 - Constructor
 - Getters
- `ProductLine` (*created inside the main method*)
 - Each `prodLine` is an `ArrayQueue`
`prodLine0 = new ArrayQueue();`
`prodLine1 = new ArrayQueue();`
 - Simulate 5 production cycles for each `prodLine`. In each cycle,
 - **Ask user what product to produce on line 0**
 - **Produce 8 products of this type and add them to `prodLine0`**
 - **Ask user what type of product to produce on line1**
 - **Produce 4 products of this type and add them to `prodLine1`**

4/8/2002

9

PS7 Hint: Stack and Stacking Area

- `Stack`
 - Each stack is an `ArrayStack`
`stack0 = new ArrayStack();`
- `class StackingArea`
 - Attributes: each `StackingArea` contains 4 stacks
 - Methods:
 - `public void addToStack(Product p)`

```
{  
    .....  
    stackXX.push(p); //How to determine which stack to add to?  
}
```
 - `public Product getFromStack()`

```
{  
    .....  
    stackYY.pop(); //How to determine which stack to get from?  
}
```
 - **To keep the heights of stacks even, rotate the number of stacks to add to and get from (e.g. starting from `stack0`, `stack1`, ... `stack3`)**
 - Hint: use an index to the next stack

4/8/2002

10

PS7 Hint: Robot and TransferCar

- class Robot
 - Pick products from production line and place them in stacking area, 4 at a time

```
void moveProduct(ArrayQueue line, StackArea s) {
    for (int i=0; i < 4; i++) {        // Move 4 stereos at a time
        //Remove a product from line
        //Add that product to s
    }
}
```

- Class TransferCar
 - Pick products based on customer order and remove them from the corresponding stacking area

```
public void pickProduct(StackArea[] s, int[] order) {
    for (int i=0; i < 6; i++)
        for (int j=0; j < order[i]; j++) {
            // Get product p from StackArea
        }
}
```

4/8/2002

11

PS7 Hint: Main Class

- Create 1 robot `r`
- Create 1 transfer car `t`
- Create an array of 6 stacking areas `s`

```
StackArea[] s= new StackArea[6];
for (int i= 0; i < 6; i++)
    s[i]= new StackArea();
```

- Create 2 production lines `prodLine0` and `prodLine1`
 - Simulate 5 production cycles for each production line
- Robot picks products from line 0,1,0; places stereos in corresponding stacking areas

```
– ProdLine0 is twice as fast as ProdLine1
```

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

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

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

4/8/2002

12

PS7 Hint: Main Class

- Prompt user for order quantities for each product type

```
int[] orderQty= new int[6];
for (int i=0; i < 6; i++) {
    String text= JOptionPane.showInputDialog("Enter qty of
product " + i + " to ship:");
    orderQty[i]= Integer.parseInt(text);
}
```

- Transfer car picks the orders from stacking areas

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