東北大學
Northeastern University

软件工程

张爽

东北大学软件学院

4.3

Data Encapsulation

& Information Hiding

Data Encapsulation

- **Example**

Design an operating system for a large mainframe computer. It has been decided that batch jobs submitted to the computer will be classified as high priority, medium priority, or low priority. There must be three queues for incoming batch jobs, one for each job type. When a job is submitted by a user, the job is added to the appropriate queue, and when the operating system decides that a job is ready to be run, it is removed from its queue and memory is allocated to it.

m_1

definition of job-queue

```
initialize_job_queue()
{
        ……
}
```

m_2

definition of job-queue

```
add_job_to_queue(job j)
{
        ……
}
```

m_3

definition of job-queue

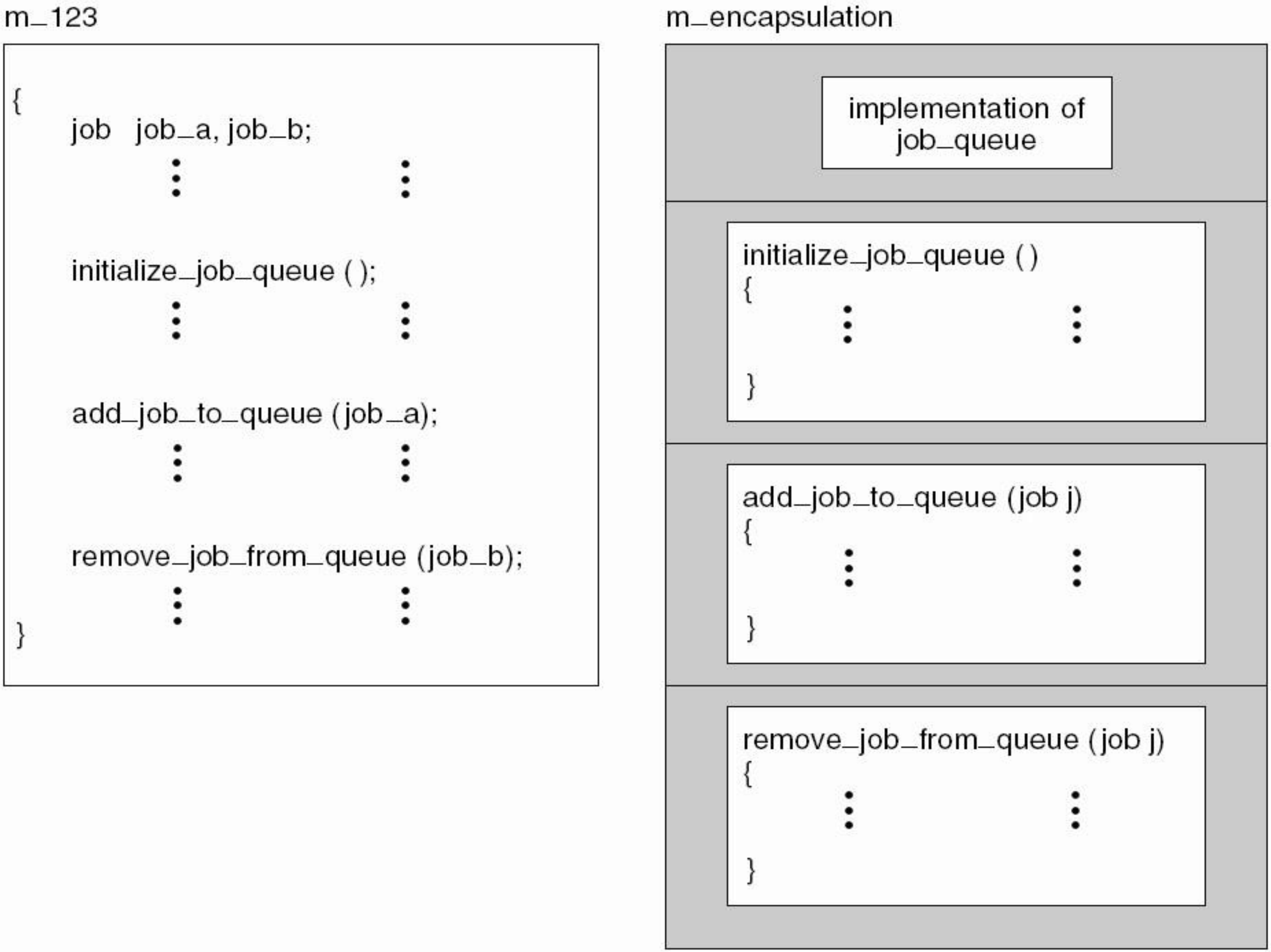
```
remove_job_from_queue(job j)
{
        ……
}
```

m_123

definition of job-queue

```
{
  job job_a, job_b;

  initialize_job_queue() {
      ……
  }
  add_job_to_queue (job_a) {
      ……
  }
  remove_job_from_queue (job_b) {
      ……
  }
}
```



m_123

```
{
      job   job_a, job_b;
                ⋮                  ⋮

      initialize_job_queue ( );
                ⋮                  ⋮

      add_job_to_queue (job_a);
                ⋮                  ⋮

      remove_job_from_queue (job_b);
                ⋮                  ⋮
}
```

m_encapsulation

implementation of job_queue

```
initialize_job_queue ( )
{
         ⋮                  ⋮

}
```

```
add_job_to_queue (job j)
{
         ⋮                  ⋮

}
```

```
remove_job_from_queue (job j)
{
         ⋮                  ⋮

}
```

Data Encapsulation

- *m_encapsulation* has *informational cohesion*.
- *m_encapsulation* is an implementation of *data encapsulation*, that is, a data structure, together with the operations to be performed on that data structure.

Advantage of Data Encapsulation

1. Data encapsulation & Development

- **Data encapsulation is an example of *abstraction***
- **Job queue example**
  - **– Data structure**
    - **job_queue**
  - **– Three operations**
    - **initialize_job_queue**
    - **add_job_to_queue**
    - **delete_job_from_queue**

Data Abstraction

Advantage of Data Encapsulation

- Data abstraction allows the designer to think at the level of the data structure and the operations performed on it, and only later be concerned with the details of how the data structure and operations are implemented.

Advantage of Data Encapsulation

2. Data encapsulation & Maintenance

- Approaching data encapsulation from the viewpoint of maintenance, a basic issue is to identify the aspects of a product likely to change and design the product to minimize the effects of future changes.

Advantage of Data Encapsulation

```
class JobQueueClass{
    public int queueLength;   // length of job queue
    public int queue[] = new int[25];
    public void initializeJobQueue(){
        queueLength = 0;
    }
    public void addJobToQueue(int jobNumber){
        ......
    }
    public void removeJobFromQueue(){
        ......
    }
}
```

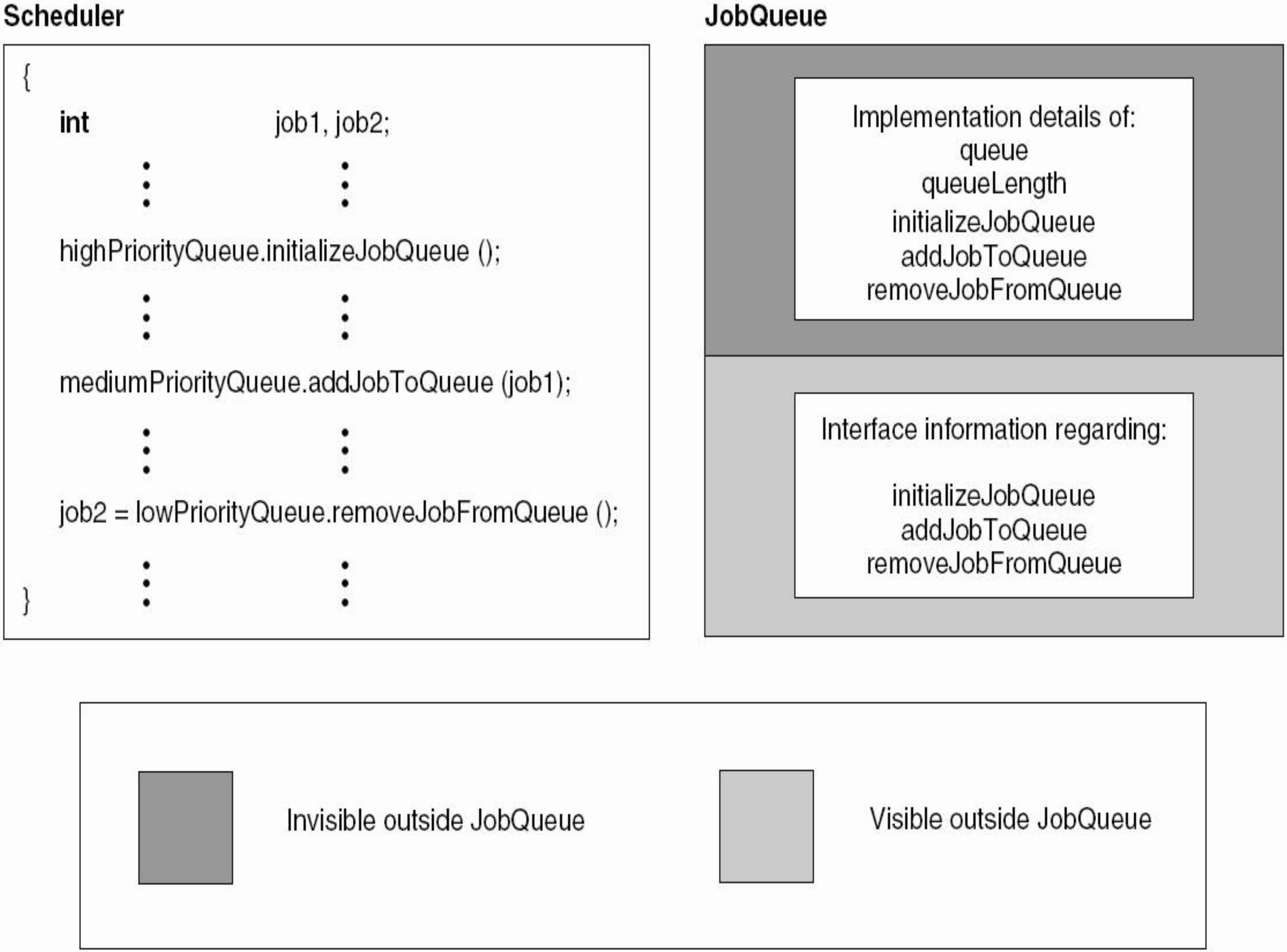
Abstract Data Type

- **Abstract data type ---- a data type together with the actions to be performed on instantiations of that data type.**

Information Hiding

- **Data abstraction**
  - **Designer thinks at level of an Abstract Data Type**
- ***information hiding* → *detail hiding***
  - **Design the modules in way that implementation details are hidden from other modules**
  - **Future change is localized**
  - **Changes cannot affect other modules**

```
class JobQueue{
        private int queueLength;   // length of job queue
        private int queue[] = new int[25];
        public void initializeJobQuieue(){
                queueLength = 0;
        }
        public void addJobToQuieue(int jobNumber){
                ......
        }
        public void removeJobFromQuieue(){
                ......
        }
}
```



Scheduler

```
{
    int                 job1, job2;
              ⋮               ⋮
    highPriorityQueue.initializeJobQueue ();
              ⋮               ⋮
    mediumPriorityQueue.addJobToQueue (job1);
              ⋮               ⋮
    job2 = lowPriorityQueue.removeJobFromQueue ();
              ⋮               ⋮
}
```

JobQueue

Implementation details of:
queue
queueLength
initializeJobQueue
addJobToQueue
removeJobFromQueue

Interface information regarding:

initializeJobQueue
addJobToQueue
removeJobFromQueue

▩ Invisible outside JobQueue

▨ Visible outside JobQueue