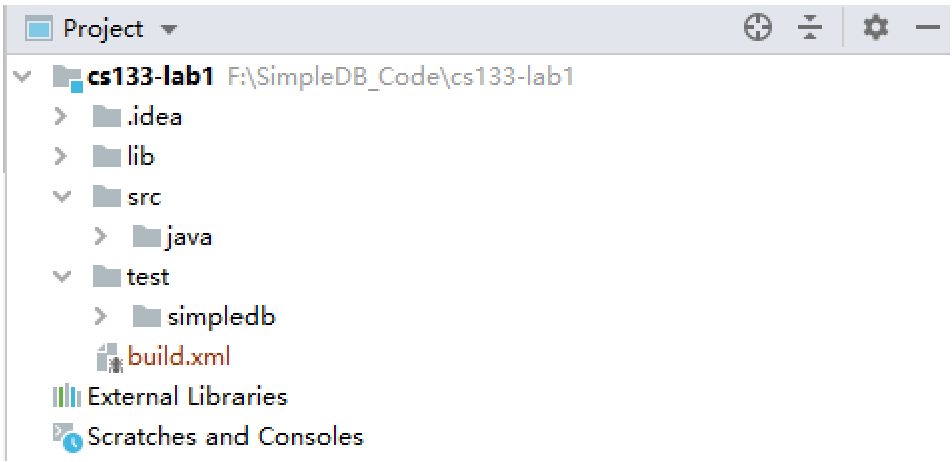
simpledb
- 环境配置
  - 使用Idea
  - 使用Eclipse
- 介绍
- Exercise 1
- Exercise 2
- Exercise 3
- Exercise 4
- Exercise 5
- Exercise 6

simpledb

环境配置

使用Idea

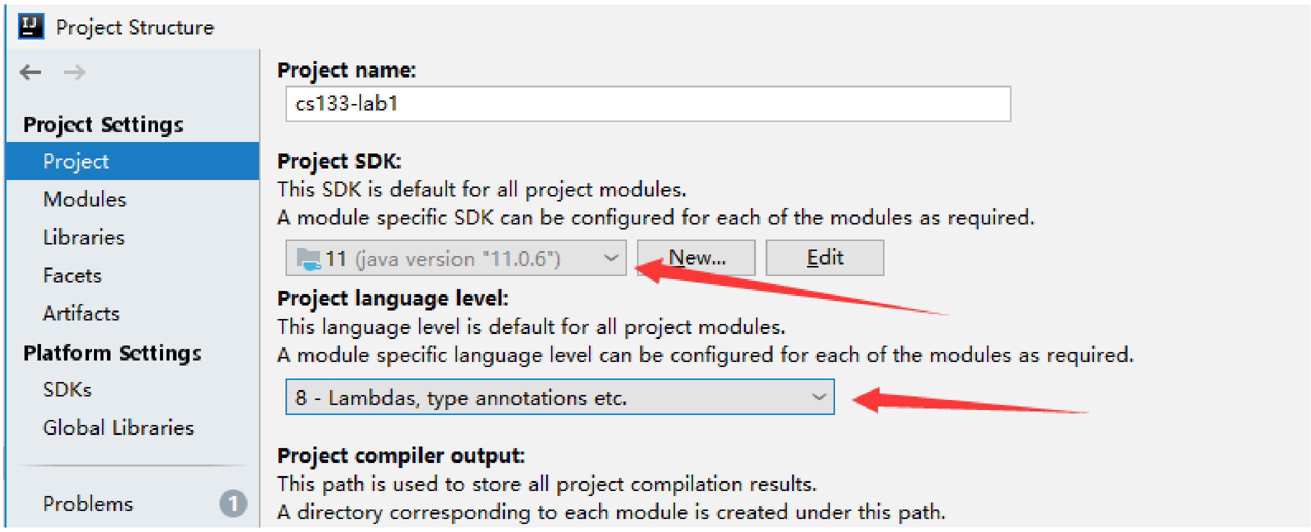
打开工程



```
Project
cs133-lab1  F:\SimpleDB_Code\cs133-lab1
  .idea
  lib
  src
    java
  test
    simpledb
  build.xml
External Libraries
Scratches and Consoles
```

`ctrl+alt+shift+s` 打开 `Project Structure`

配置 SDK



Project Structure

Project name:

cs133-lab1

Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.

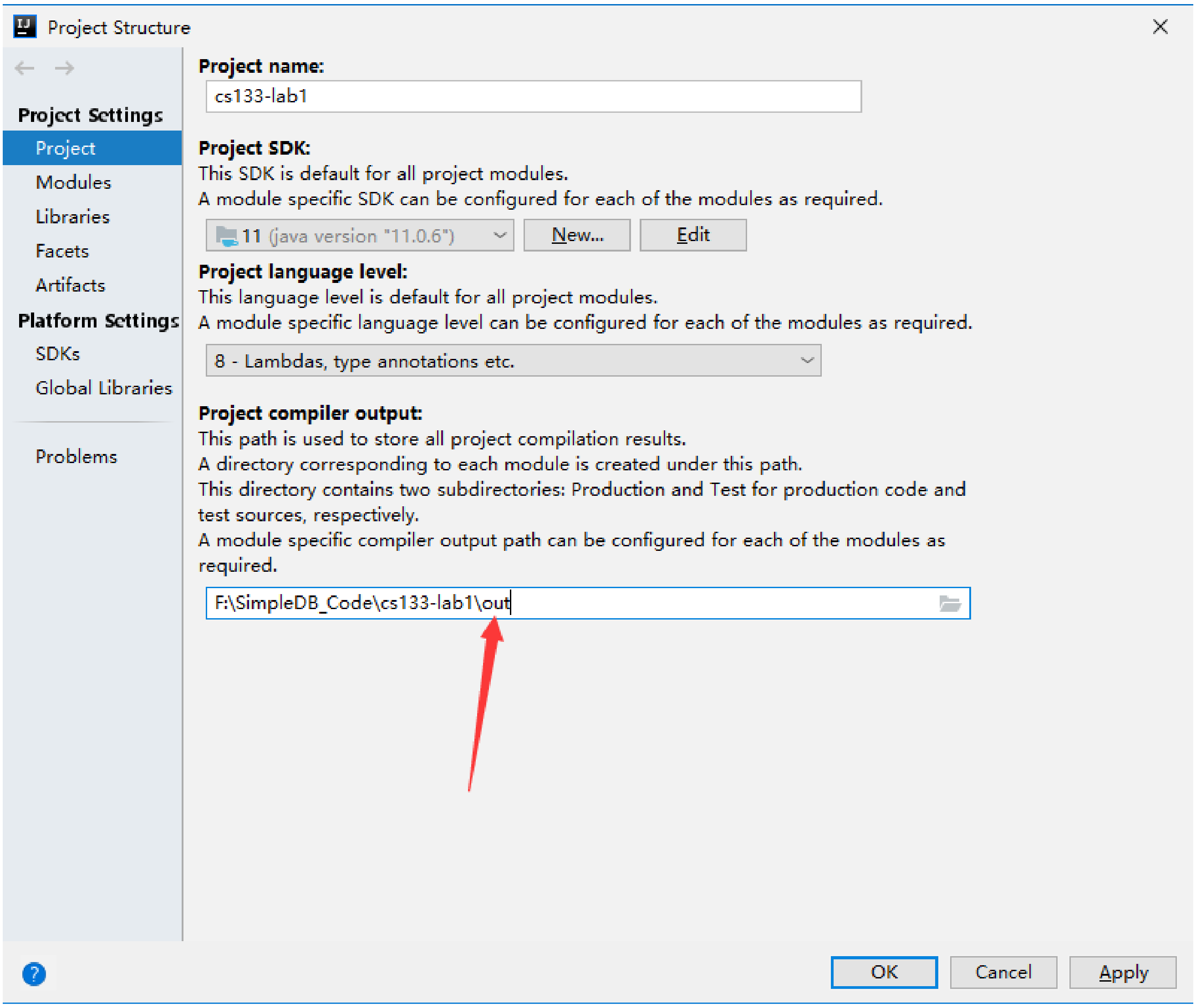
11 (java version "11.0.6")

Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.

8 - Lambdas, type annotations etc.

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.

然后配置 `Project compiler output`，路径我们配置成 当前项目路径\out 即可



Project Structure

Project name:

cs133-lab1

Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.

11 (java version "11.0.6")

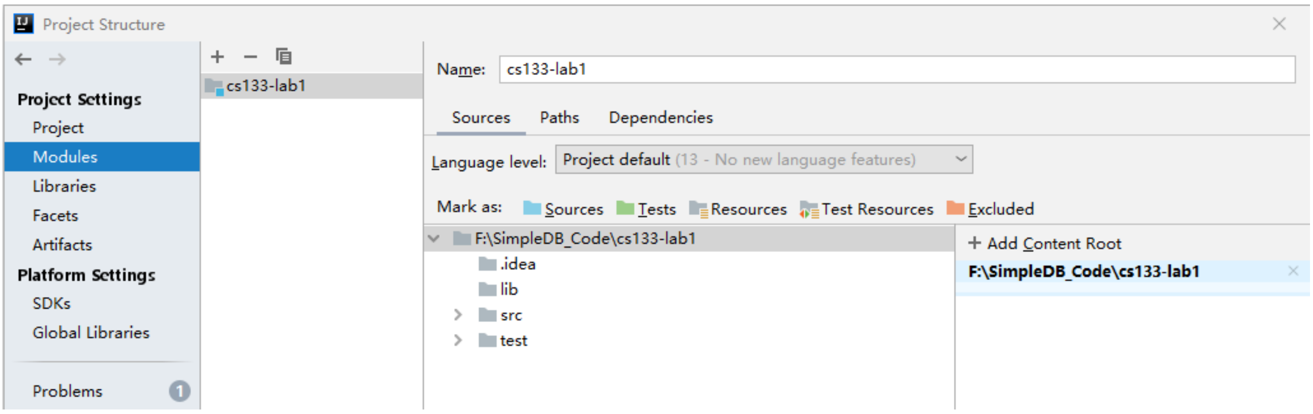
Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.

8 - Lambdas, type annotations etc.

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.

F:\SimpleDB_Code\cs133-lab1\out

切换到 Modules



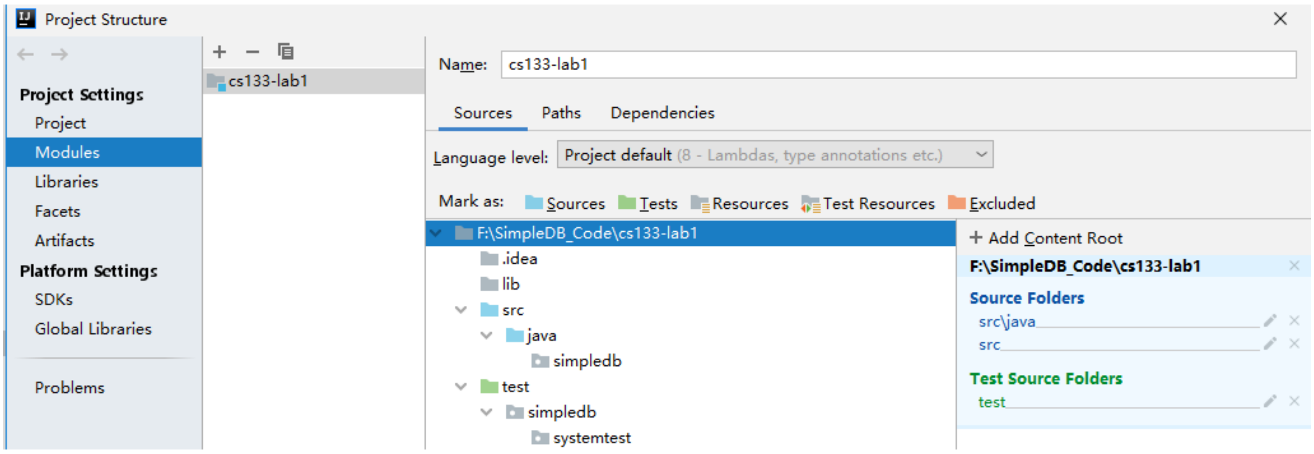
Project Structure

Name: cs133-lab1

Language level: Project default (13 - No new language features)

F:\SimpleDB_Code\cs133-lab1
- .idea
- lib
- src
- test

如下配置文件夹属性



Project Structure

Name: cs133-lab1

Language level: Project default (8 - Lambdas, type annotations etc.)

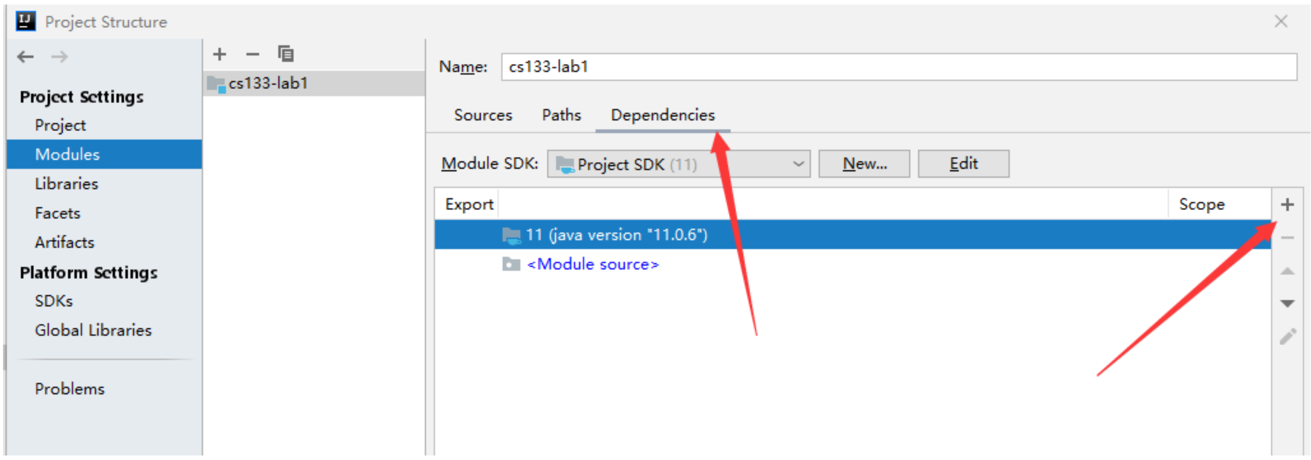
F:\SimpleDB_Code\cs133-lab1
- .idea
- lib
- src
  - java
    - simpledb
- test
  - simpledb
    - systemtest

F:\SimpleDB_Code\cs133-lab1

Source Folders
- src\java
- src

Test Source Folders
- test

然后切换到 Denpendencies，添加 Project 依赖的包



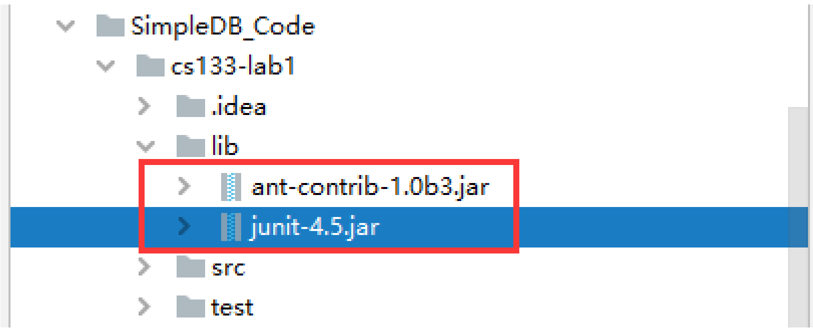
Project Structure

Name: cs133-lab1

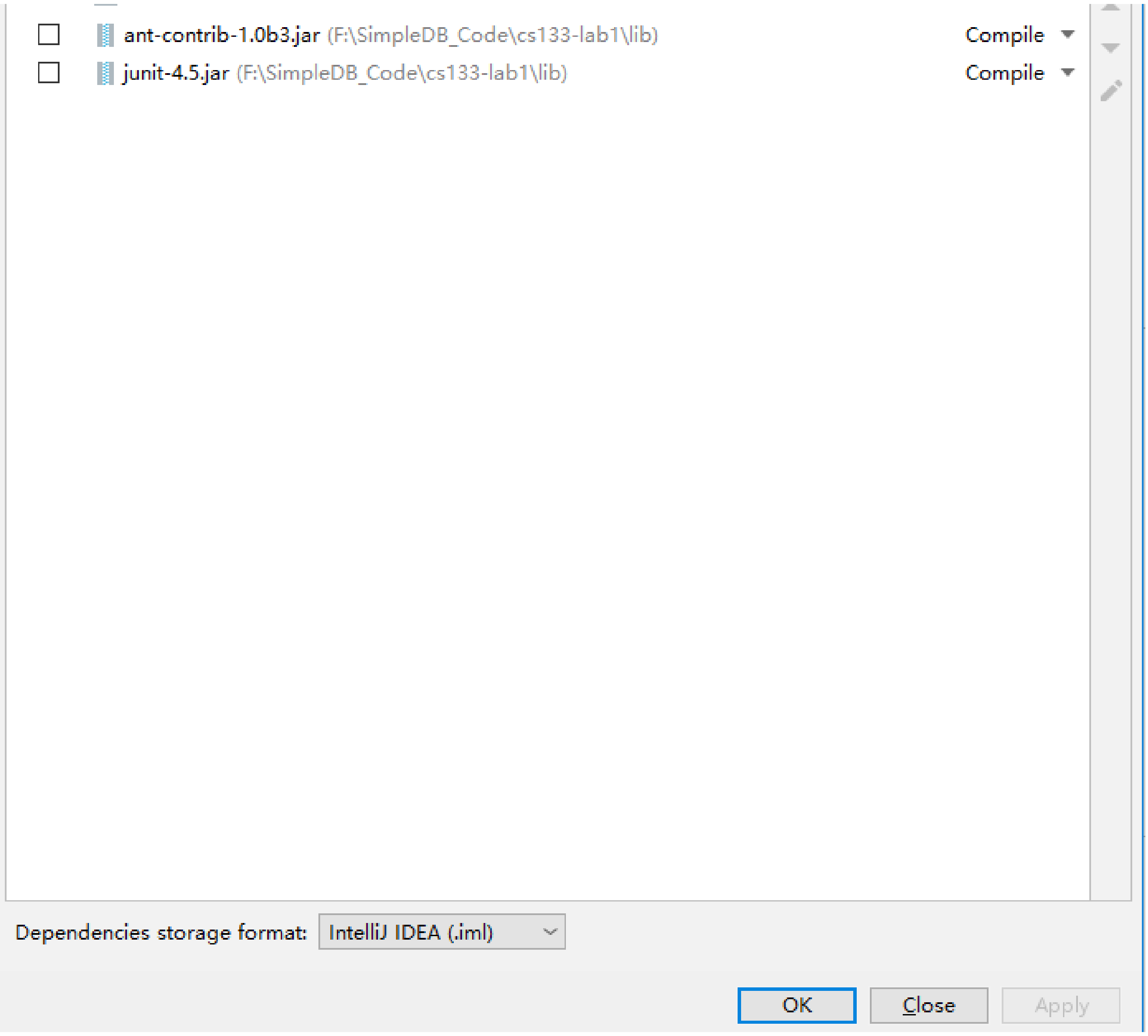
Module SDK: Project SDK (11)

Export		Scope
	11 (java version "11.0.6")	
	<Module source>	

依赖包在 lib 文件夹下



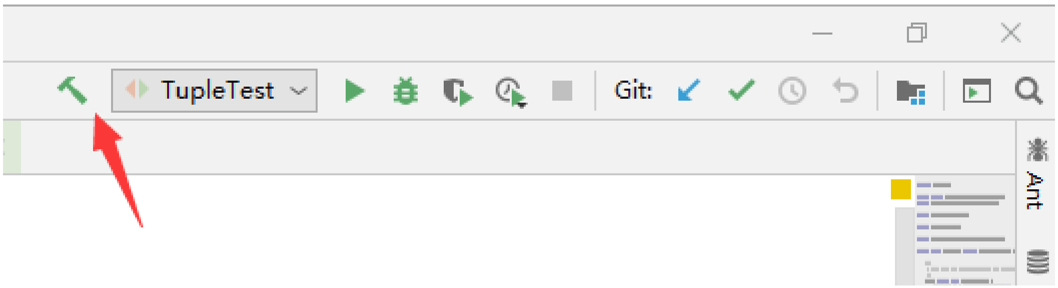
```
SimpleDB_Code
  cs133-lab1
    .idea
    lib
      ant-contrib-1.0b3.jar
      junit-4.5.jar
    src
    test
```



□ ant-contrib-1.0b3.jar (F:\SimpleDB_Code\cs133-lab1\lib) Compile

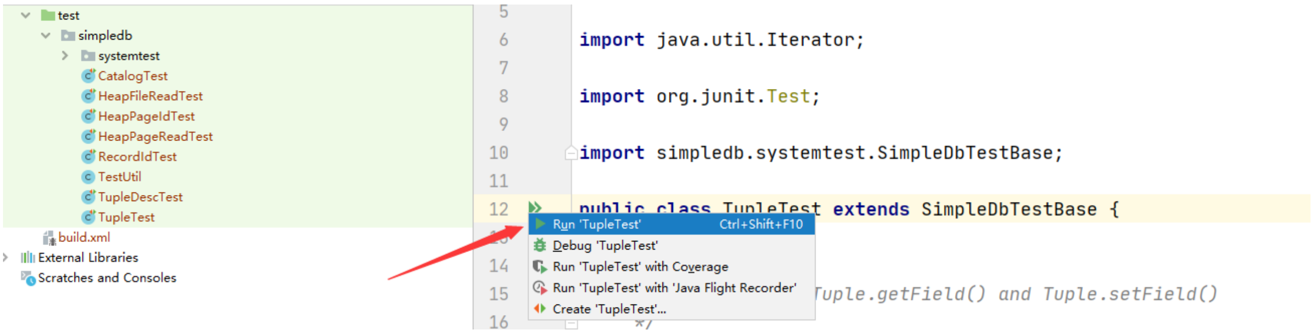
□ junit-4.5.jar (F:\SimpleDB_Code\cs133-lab1\lib) Compile

Dependencies storage format: IntelliJ IDEA (.iml)

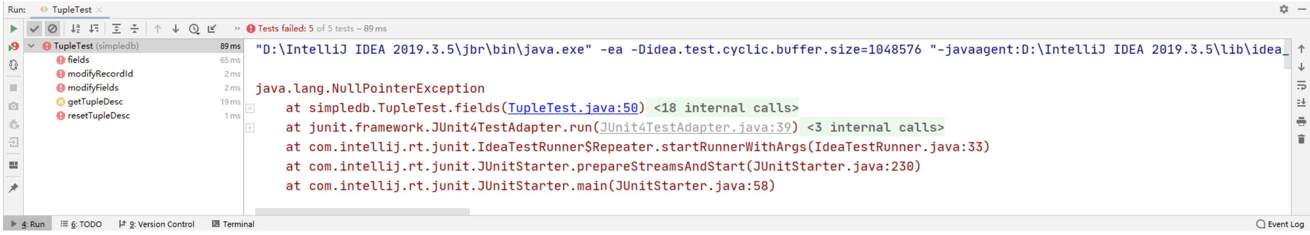


点击图标或 `ctrl+F9` 构建项目，构建完毕后重启 IDEA

打开 `test` 文件夹下的 `TupleTest`，运行测试样例，如果测试样例能够启动，则配置成功



```java

import java.util.Iterator;

import org.junit.Test;

import simpledb.systemtest.SimpleDbTestBase;

public class TupleTest extends SimpleDbTestBase {

        Tuple.getField() and Tuple.setField()
     */
```



Tests failed: 5 of 5 tests – 89 ms

- TupleTest (simpledb) 89 ms
  - fields 65 ms
  - modifyRecordId 2 ms
  - modifyFields 2 ms
  - getTupleDesc 19 ms
  - resetTupleDesc 1 ms

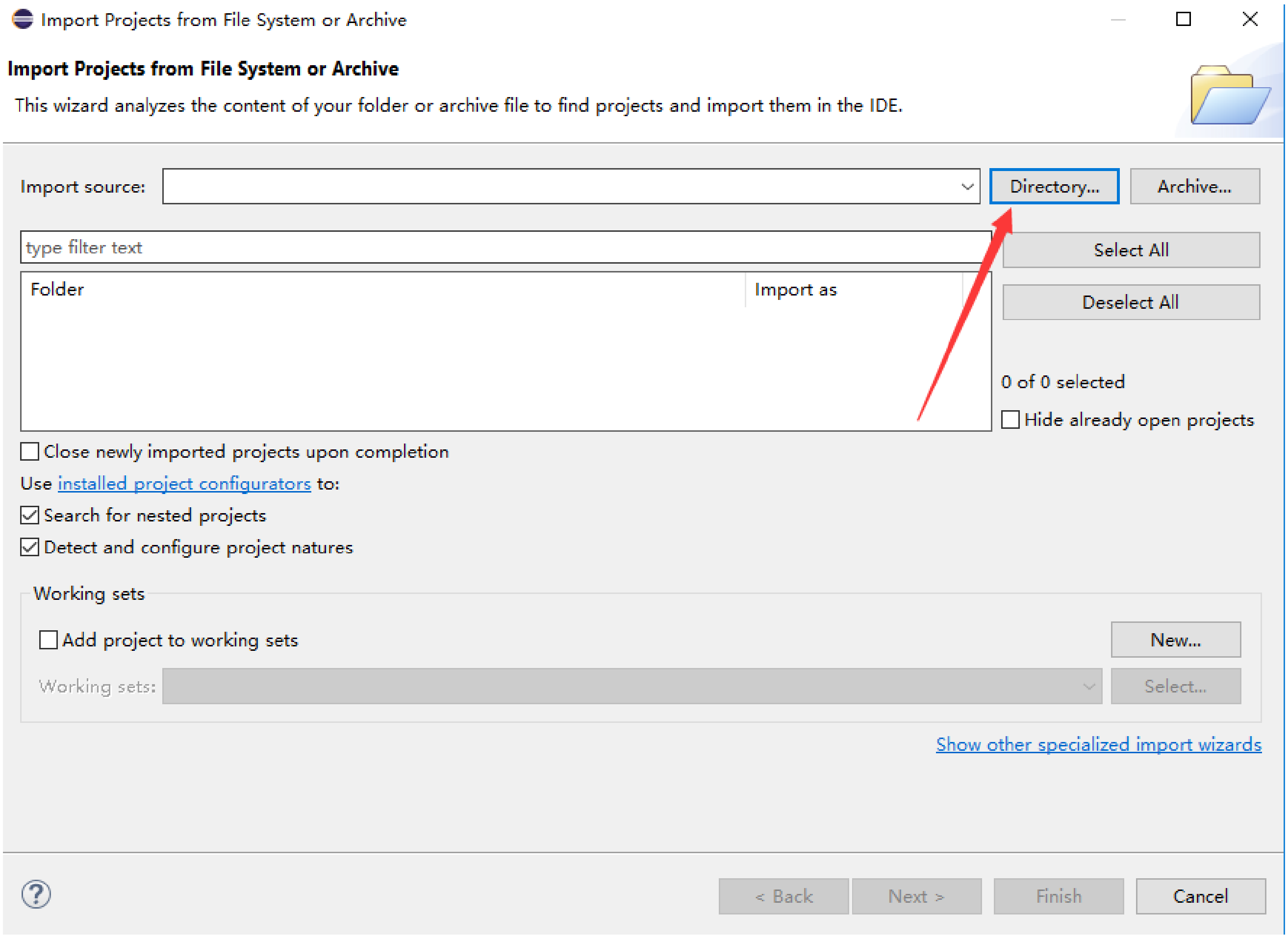
```
"D:\IntelliJ IDEA 2019.3.5\jbr\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:D:\IntelliJ IDEA 2019.3.5\lib\idea_

java.lang.NullPointerException
    at simpledb.TupleTest.fields(TupleTest.java:50) <18 internal calls>
    at junit.framework.JUnit4TestAdapter.run(JUnit4TestAdapter.java:39) <3 internal calls>
    at com.intellij.rt.junit.IdeaTestRunner$Repeater.startRunnerWithArgs(IdeaTestRunner.java:33)
    at com.intellij.rt.junit.JUnitStarter.prepareStreamsAndStart(JUnitStarter.java:230)
    at com.intellij.rt.junit.JUnitStarter.main(JUnitStarter.java:58)
```

当出现以上结果时，环境配置完成，可以开始写代码了

使用Eclipse

最新版的eclipse

点击File -> Open Projects from File System...



Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source:

type filter text

Folder | Import as

0 of 0 selected

□ Hide already open projects

□ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

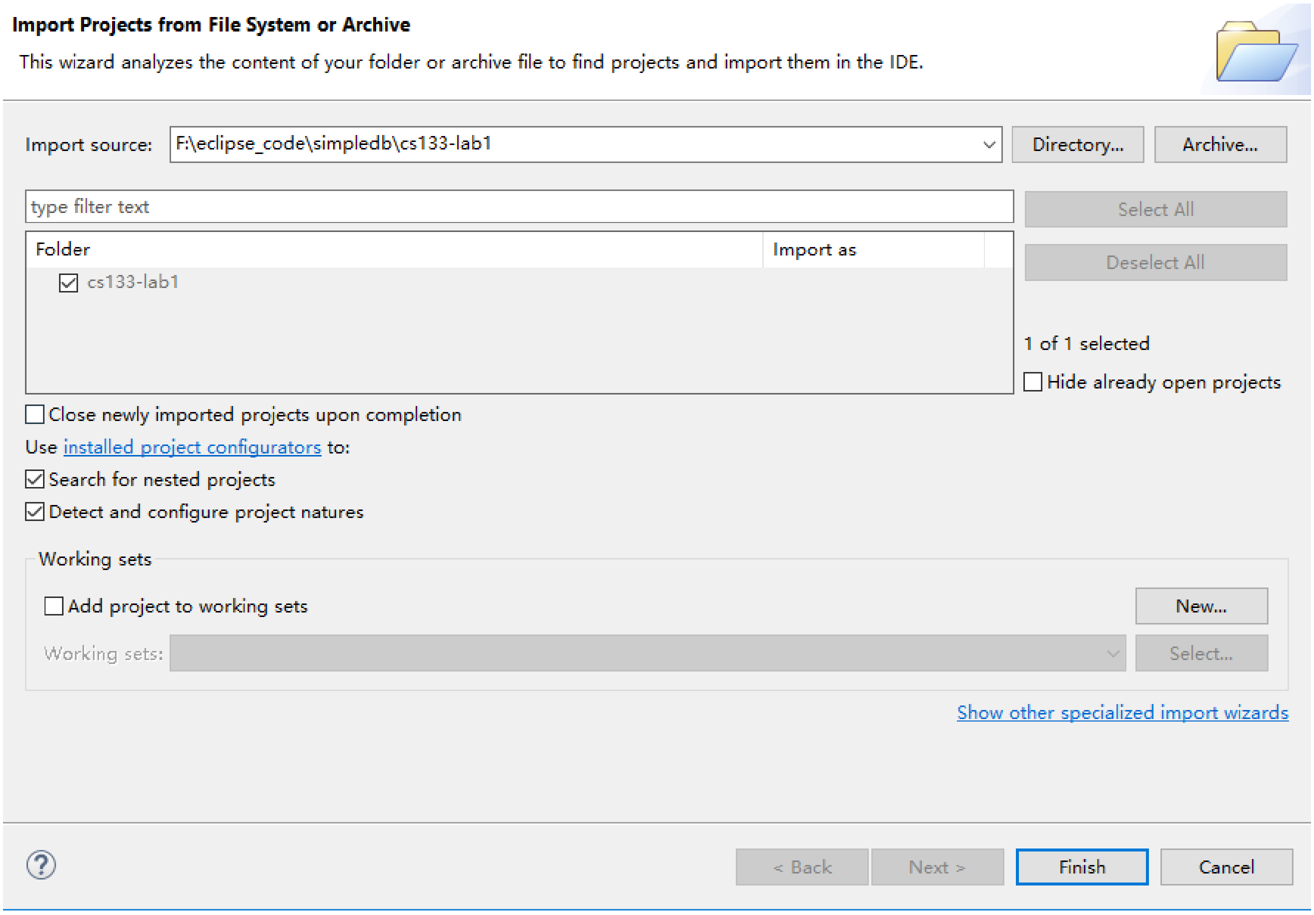
Working sets

□ Add project to working sets

Working sets:

Show other specialized import wizards

选择解压后的文件夹，finish即可



Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: F:\eclipse_code\simpledb\cs133-lab1

Folder	Import as
☑ cs133-lab1	

1 of 1 selected

□ Hide already open projects

□ Close newly imported projects upon completion

Use installed project configurators to:

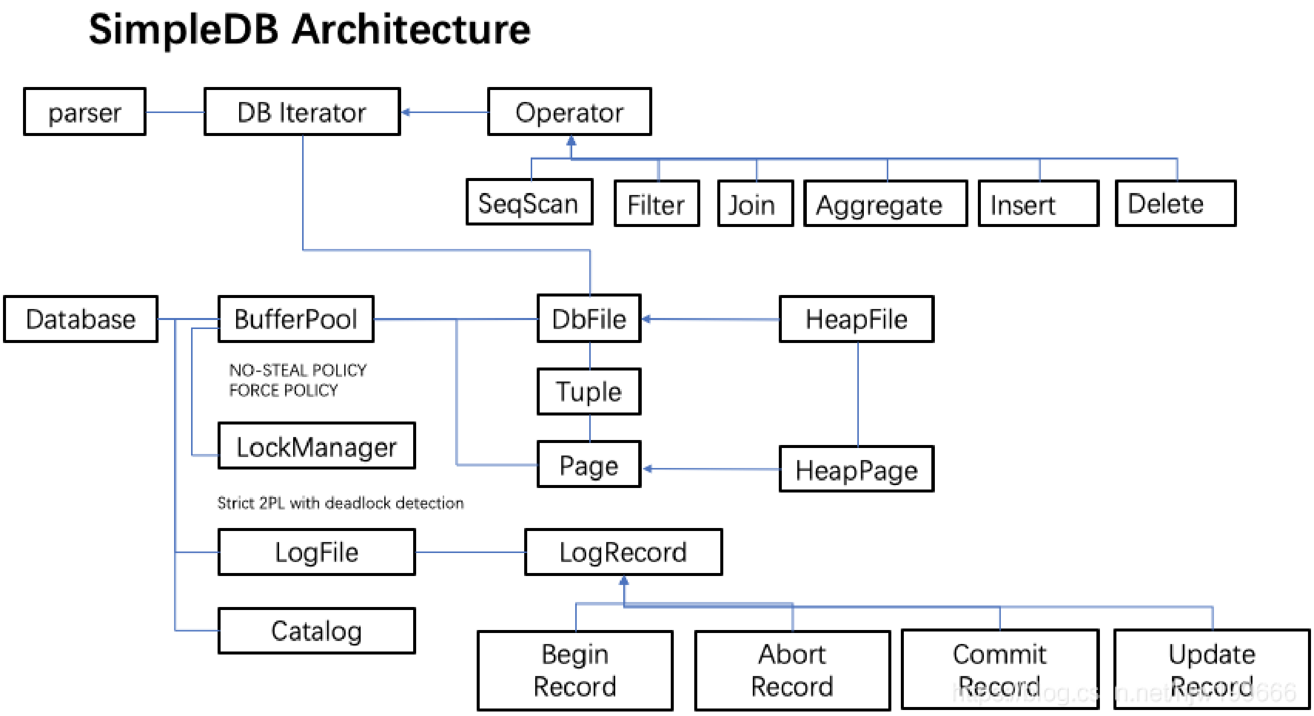
☑ Search for nested projects

☑ Detect and configure project natures

Working sets

□ Add project to working sets

Working sets:



介绍

simpledb lab1 出自 MIT 6.830 课程，总共6个 exercise 需要完成，前3个 exercise 必须要完成，后三个 exercise 可选做。

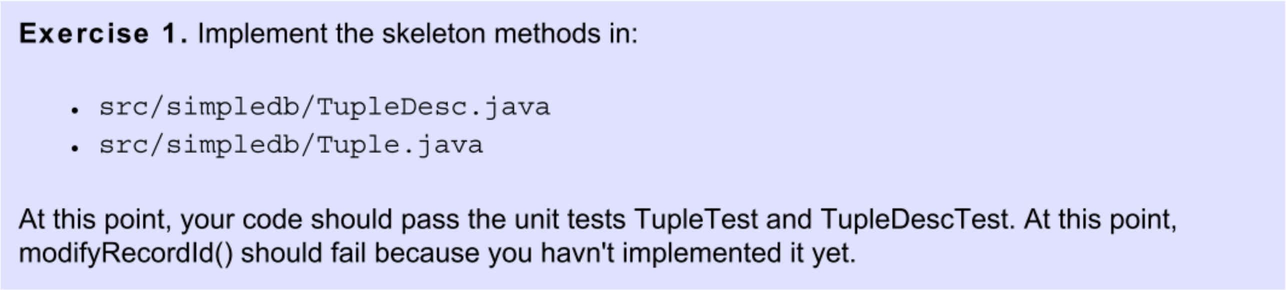
project 的大部分代码已经写好，我们需要在带有 `// some code goes here` 样式的地方补全代码，然后通过测试样例

lab1 中总共有 7 个 java 文件需要补全：

- Tuple
- TupleDesc
- Catalog
- BufferPool
- HeapFile
- HeapPage
- SeqScan

lab1 的 PDF 里面有详细的说明，下面简单介绍一下每个练习要完成的工作

Exercise 1



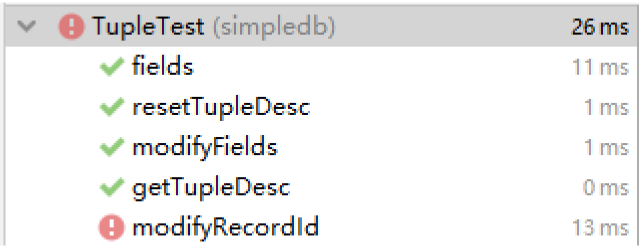
Exercise 1. Implement the skeleton methods in:

- `src/simpledb/TupleDesc.java`
- `src/simpledb/Tuple.java`

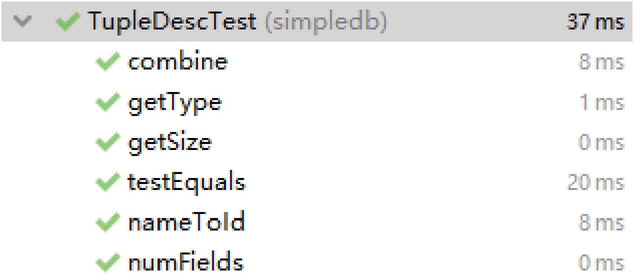
At this point, your code should pass the unit tests TupleTest and TupleDescTest. At this point, modifyRecordId() should fail because you havn't implemented it yet.

- TupleDesc
- Tuple

通过：

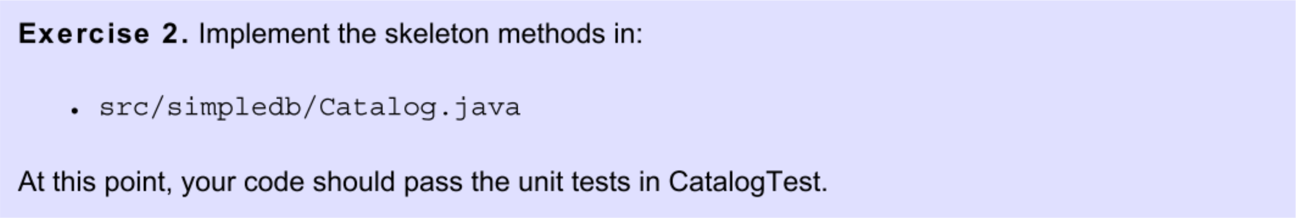


```
TupleTest (simpledb)      26 ms
    fields                11 ms
    resetTupleDesc         1 ms
    modifyFields           1 ms
    getTupleDesc           0 ms
    modifyRecordId        13 ms
```



```
TupleDescTest (simpledb)  37 ms
    combine                8 ms
    getType                1 ms
    getSize                0 ms
    testEquals            20 ms
    nameToId               8 ms
    numFields              0 ms
```

Exercise 2



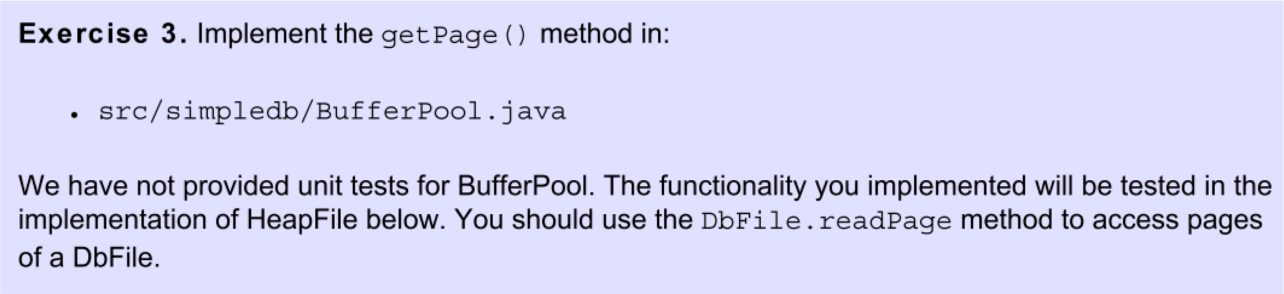
Exercise 2. Implement the skeleton methods in:

- `src/simpledb/Catalog.java`

At this point, your code should pass the unit tests in CatalogTest.

Catalog管理数据库表相关信息

Exercise 3



Exercise 3. Implement the `getPage()` method in:

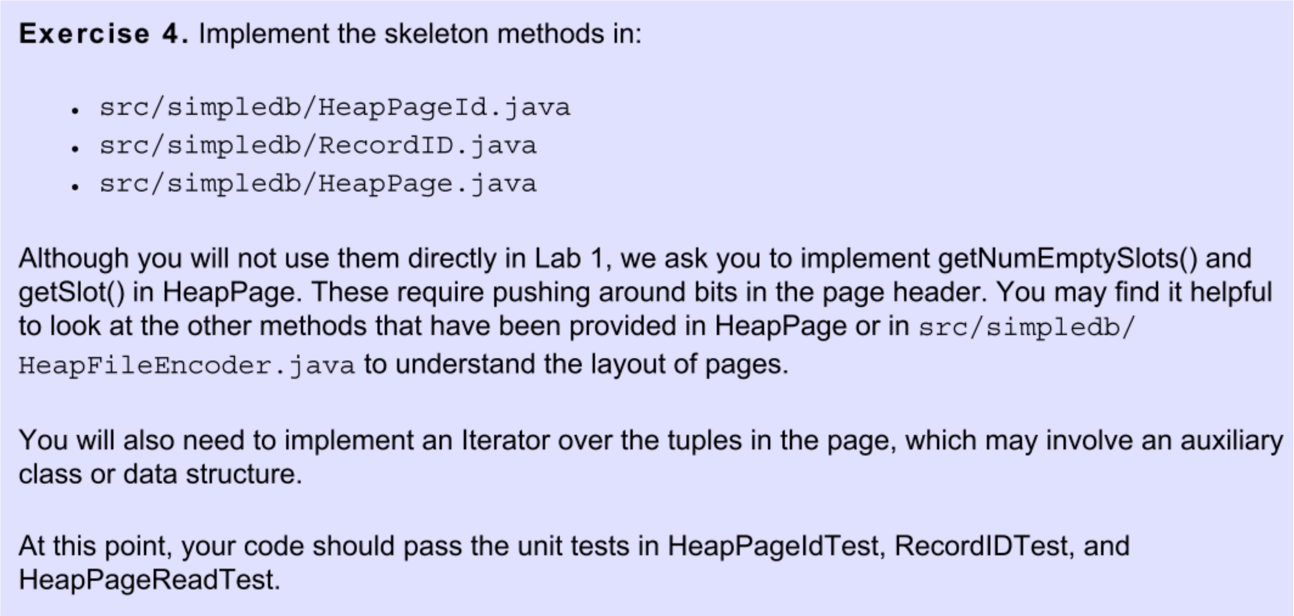
- `src/simpledb/BufferPool.java`

We have not provided unit tests for BufferPool. The functionality you implemented will be tested in the implementation of HeapFile below. You should use the `DbFile.readPage` method to access pages of a DbFile.

BufferPool没有测试样例

Catalog和BufferPool由Database这个单例类持有，用于管理数据库

Exercise 4



Exercise 4. Implement the skeleton methods in:

- `src/simpledb/HeapPageId.java`
- `src/simpledb/RecordID.java`
- `src/simpledb/HeapPage.java`

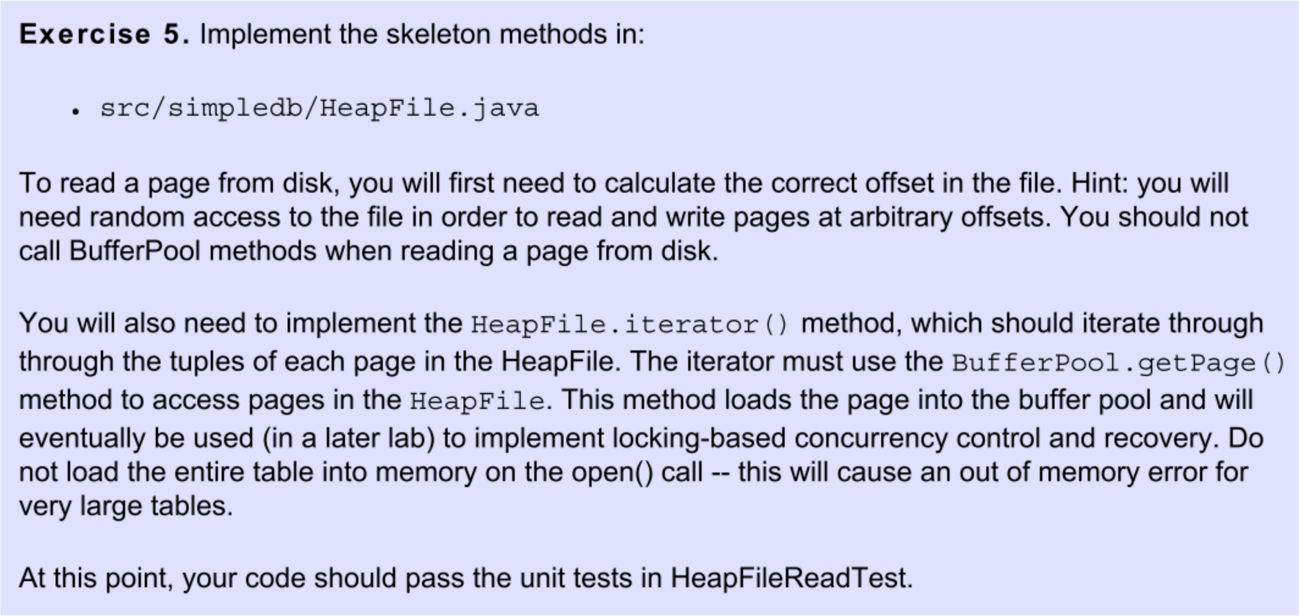
Although you will not use them directly in Lab 1, we ask you to implement getNumEmptySlots() and getSlot() in HeapPage. These require pushing around bits in the page header. You may find it helpful to look at the other methods that have been provided in HeapPage or in `src/simpledb/HeapFileEncoder.java` to understand the layout of pages.

You will also need to implement an Iterator over the tuples in the page, which may involve an auxiliary class or data structure.

At this point, your code should pass the unit tests in HeapPageIdTest, RecordIDTest, and HeapPageReadTest.

完成Page相关的三个类

Exercise 5



Exercise 5. Implement the skeleton methods in:

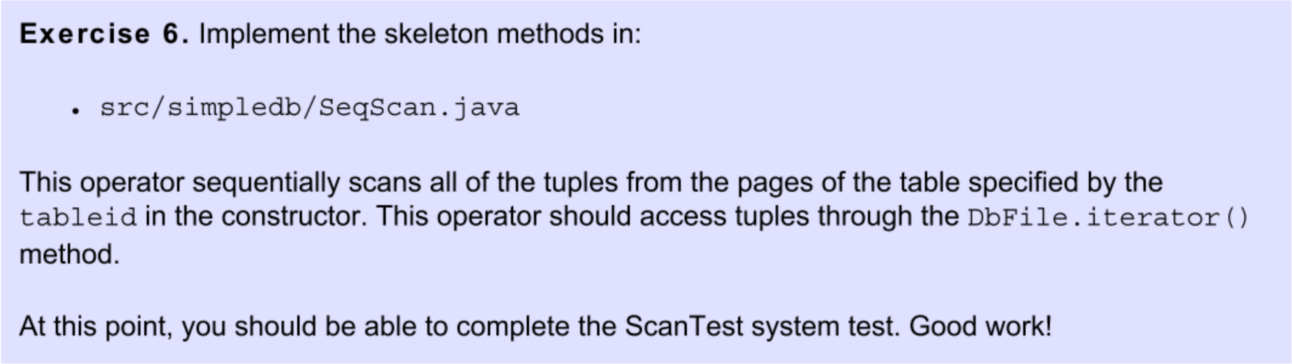
- `src/simpledb/HeapFile.java`

To read a page from disk, you will first need to calculate the correct offset in the file. Hint: you will need random access to the file in order to read and write pages at arbitrary offsets. You should not call BufferPool methods when reading a page from disk.

You will also need to implement the `HeapFile.iterator()` method, which should iterate through through the tuples of each page in the HeapFile. The iterator must use the `BufferPool.getPage()` method to access pages in the `HeapFile`. This method loads the page into the buffer pool and will eventually be used (in a later lab) to implement locking-based concurrency control and recovery. Do not load the entire table into memory on the open() call -- this will cause an out of memory error for very large tables.

At this point, your code should pass the unit tests in HeapFileReadTest.

Exercise 6



Exercise 6. Implement the skeleton methods in:

- `src/simpledb/SeqScan.java`

This operator sequentially scans all of the tuples from the pages of the table specified by the `tableid` in the constructor. This operator should access tuples through the `DbFile.iterator()` method.

At this point, you should be able to complete the ScanTest system test. Good work!