题目 1：Using File I/O in the Bright Fresh Milk System

Prerequisites, Goals, and Outcomes

Prerequisites: Before you begin this exercise, you need mastery of the following:

- *Java API*
    - Knowledge of the class `StringTokenizer`
- *File I/O*
    - Knowledge of file I/O
        - How to read data from a file
        - How to write data to a file

Goals: Reinforce your ability to use file I/O

Outcomes: You will master the following skills:

- Produce applications that read data from a file and parse it
- Produce applications that write data to a file

Background

In this assignment, you will create another version of the Bright Fresh Milk System. In previous versions, the data for the product catalog was hard-coded in the application. In this version, the data will be loaded from a file. Also, the user will be able to write the sales information to a file in one of three formats: plain text, HTML, or XML. Part of the work has been done for you and is provided in the student archive. You will implement the code that loads the product catalog and persists the sales information.

Description

The Bright Fresh Milk System sells four types of products: Pure Milk, Jelly, Yogurt, and Milk Drink. A file called catalog.dat stores the product data:

- *catalog.dat*. File with product data

Every line in catalog.dat contains exactly one product.

1）A line for a Pure Milk product has the following format:

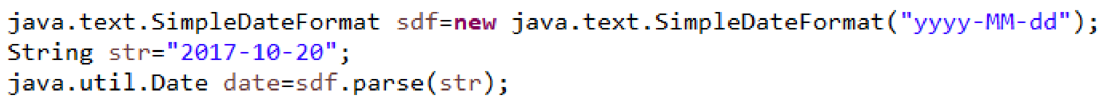
PureMilk_code_description_price_productionDate_shelfLife_countryOfOrigin_butterfat_protein

where:

- "*PureMilk*" is a prefix that indicates the line type.
- *code* is a string that represents the code of the *PureMilk*.
- *description* is a string that represents the description of the *PureMilk*.
- *price* is a BigDecimal that represents the price of the *PureMilk*.
- *productionDate* is a Date that represents the production Date of the PureMilk.
- *shelfLife* is a string that represents the *shelf life* of the *PureMilk*.
- *countryOfOrigin* is a string that represents the *origin* of the *PureMilk*.
- *butterfat* is a string that represents the *butterfat* of the *PureMilk*.
- *protein* is a string that represents the *protein* of the *PureMilk*.

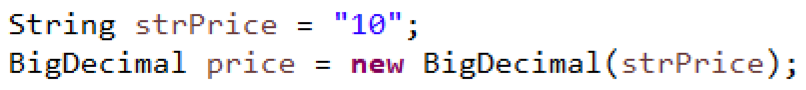
Tips：

- *java.lang.String to java.util.Date*



```
java.text.SimpleDateFormat sdf=new java.text.SimpleDateFormat("yyyy-MM-dd");
String str="2017-10-20";
java.util.Date date=sdf.parse(str);
```

- *java.lang.String to java.math.BigDecimal*



```
String strPrice = "10";
BigDecimal price = new BigDecimal(strPrice);
```

The fields are delimited by an underscore (_). You can assume that the fields themselves do not contain any underscores.

2）A line for a *Jelly* has the following format:

Jelly_code_description_price_productionDate_shelfLife_type_diluteConcentration

where:

- "*Jelly*" is a prefix that indicates the line type.
- *code* is a string that represents the code of the *Jelly*.
- *description* is a string that represents the description of the *Jelly*.

- *price* is a BigDecimal that represents the price of the *Jelly*.
- *productionDate* is a Date that represents the production Date of the *Jelly*.
- *shelfLife* is a string that represents the *shelf life* of the *Jelly*.
- *type* is a string that represents the *origin* of the *Jelly*.
- *diluteConcentration* is a string that represents the *dilute Concentration* of the *Jelly*.

The fields are delimited by an underscore (_). You can assume that the fields themselves do not contain any underscores.

3）A line for a *Yogurt* has the following format:

Yogurt_code_description_price_productionDate_shelfLife_flavor

where:

- "*Yogurt*" is a prefix that indicates the line type.
- *code* is a string that represents the code of the *Yogurt*.
- *description* is a string that represents the description of the *Yogurt*.
- *price* is a BigDecimal that represents the price of the *Yogurt*.
- *productionDate* is a Date that represents the production Date of the *Yogurt*.
- *shelfLife* is a string that represents the *shelf life* of the *Yogurt*.
- *flavor* is a string that represents the *flavor* of the *Yogurt*.

The fields are delimited by an underscore (_). You can assume that the fields themselves do not contain any underscores.

4）A line for a *MilkDrink* has the following format:

MilkDrink_code_description_price_productionDate_shelfLife_flavor_sugar

where:

- "*MilkDrink*" is a prefix that indicates the line type.
- *code* is a string that represents the code of the *MilkDrink*.
- *description* is a string that represents the description of the *MilkDrink*.
- *price* is a BigDecimal that represents the price of the *MilkDrink*.
- *productionDate* is a Date that represents the production Date of the *MilkDrink*.

- *shelfLife* is a string that represents the *shelf life* of the *MilkDrink*.
- *flavor* is a string that represents the *flavor* of the *MilkDrink*.
- *sugar* is a string that represents the *sugar* of the *MilkDrink*.

The fields are delimited by an underscore (_). You can assume that the fields themselves do not contain any underscores.

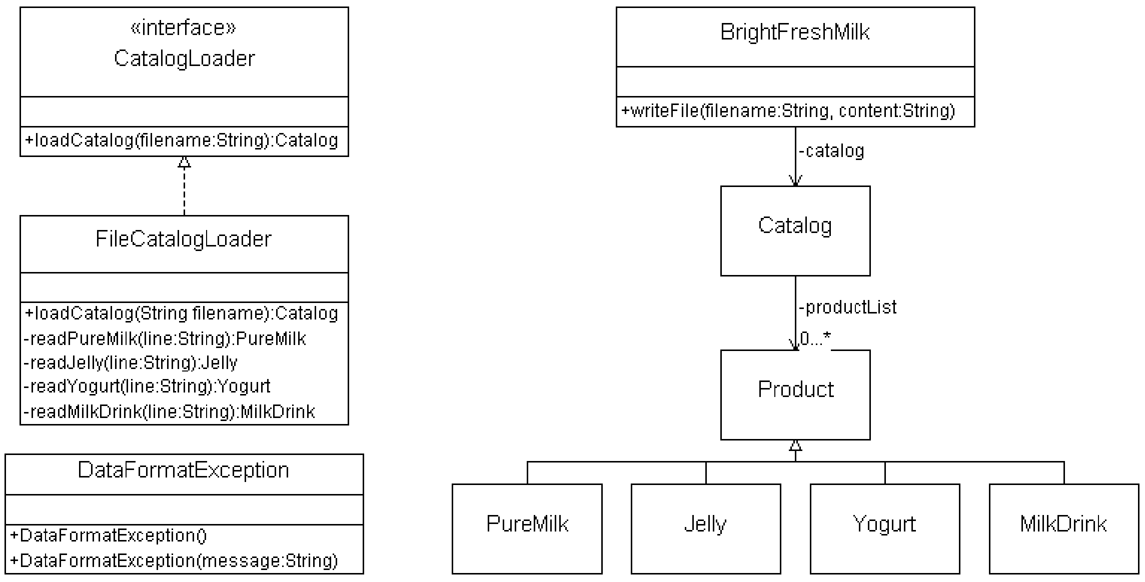

«interface»
CatalogLoader
+loadCatalog(filename:String):Catalog
FileCatalogLoader
+loadCatalog(String filename):Catalog
-readPureMilk(line:String):PureMilk
-readJelly(line:String):Jelly
-readYogurt(line:String):Yogurt
-readMilkDrink(line:String):MilkDrink
DataFormatException
+DataFormatException()
+DataFormatException(message:String)
BrightFreshMilk
+writeFile(filename:String, content:String)
-catalog
Catalog
-productList
0...*
Product
PureMilk
Jelly
Yogurt
MilkDrink

Figure 1 *Portion of Bright Fresh Milk System class diagram*

In this assignment, you will implement `FileCatalogloader` and complete the implementation of `BrightFreshMilk`.

Interface `CatalogLoader`

The interface `CatalogLoader` declares a method for producing a product catalog. A complete implementation of this interface is provided in the student archive.

Method:

- *`Catalog loadCatalog(String fileName)`*

- *throws FileNotFoundException,*
- *IOException,*
- *DataFormatException*
- *ParseException*

Loads the information in the specified file into a product catalog and returns the catalog.

Class `DataFormatException`

This exception is thrown when a line in the file being parsed has errors:

- The line does not have the expected number of tokens.

A complete implementation of this class is provided in the student archive.

Class `FileCatalogLoader`

The class `FileCatalogLoader` implements interface `CatalogLoader`. It is used to obtain a product catalog from a file. You should implement this class from scratch:

Methods:

- *private PureMilk readPureMilk (String line)*
- *throws DataFormatException, ParseException*

This method reads a line of *PureMilk* data. It uses the class `StringTokenizer` to extract the accessory data in the specified line. If the line is error free, this method returns a *PureMilk* object that encapsulates the accessory data. this method throws `DataFormatException`, *ParseException* that contains the line of malformed data.

- *private Jelly readJelly(String line)*
- *throws DataFormatException, ParseException*

This method reads a line of *Jelly* data. It uses the class `StringTokenizer` to extract the accessory data in the specified line. If the line is error free, this method returns a *Jelly* object that encapsulates the accessory data. this method throws DataFormatException, *ParseException* that contains the line of malformed data.

- *private Yogurt readYogurt(String line)*
- *throws DataFormatException, ParseException*

This method reads a line of *Yogurt* data. It uses the class StringTokenizer to extract the accessory data in the specified line. If the line is error free, this method returns a *Yogurt* object that encapsulates the accessory data. this method throws DataFormatException, *ParseException* that contains the line of malformed data.

- *private MilkDrink readMilkDrink(String line)*
- *throws DataFormatException, ParseException*

This method reads a line of *MilkDrink* data. It uses the class StringTokenizer to extract the accessory data in the specified line. If the line is error free, this method returns a *MilkDrink* object that encapsulates the accessory data. this method throws DataFormatException, *ParseException* that contains the line of malformed data.

- *public Catalog loadCatalog(String filename)*
- *throws FileNotFoundException,*
- *IOException,*
- *DataFormatException*
- *ParseException*

This method loads the information in the specified file into a product catalog and returns the catalog. It begins by opening the file for reading. It then proceeds to read and process each line in the file. The method String.startsWith is used to determine the line type:

  - If the line type is "PureMilk", the method readPureMilk is invoked.

    - If the line type is "Jelly", the method `readJelly` is invoked.
    - If the line type is "Yogurt", the method `readYogurt` is invoked.
    - If the line type is "MilkDrink", the method `readMilkDrink` is invoked.

After the line is processed, `loadCatalog` adds the product (PureMilk, Jelly, Yogurt or MilkDrink) to the catalog. When all the lines in the file have been processed, `load` returns the catalog to the calling method.

This method can throw the following exceptions:

- `FileNotFoundException` — if the specified file does not exist.
- `IOException` — if there is an error reading the information in the specified file.
- `DataFormatException` — if a line in the file has errors (the exception should contain the line of malformed data).
- `ParseException` — `if the date contains badly-formed data.`

Class `BrightFreshMilk`

A partial implementation of class `BrightFreshMilk` is provided in the student archive. You should implement `writeFile`, a method that writes sales information to a file:

- *`private void writeFile(String fileName, String content)`*
- *`    throws IOException`*

  This method creates a new file with the specified name, writes the specified string to the file, and then closes the file.

Class `TestFileCatalogLoader`

This class is a test driver for `FileCatalogLoader`. A complete implementation is included in the student archive. You should use this class to test your implementation of `FileCatalogLoader`.

Files

The following files are needed to complete this assignment:

- *第一题所需文件*— Download this file. This archive contains the following:
  - Class files
    - *Catalog.class*
    - *CurrentOrder.class*
    - *Jelly.class*
    - *MilkDrink.class*
    - *OrderItem.class*
    - *Product.class*
    - *PureMilk.class*
    - *Sale.class*
    - *Yogurt.class*
    - *SalesFormatter.class*
    - *PlainTextSalesFormatter.class* *use unit3*
    - *HTMLSalesFormatter.class* *use unit3*
    - *XMLSalesFormatter.class* *use unit3*
  - Java files
    - *CatalogLoader.java*. A complete implementation
    - *DataFormatException.java*. A complete implementation
    - *TestFileCatalogLoader.java*. A complete implementation
    - *BrightFreshMilk.java*. Use this template to complete your implementation.
  - Data files for the test driver
    - *catalog.dat*. A file with product information
    - *empty.dat*. An empty file

Tasks

Implement the class `FileCatalogLoader` and the

method `BrightFreshMilk.writeFile`. Document using Javadoc and follow

Sun's code conventions. The following steps will guide you through this assignment. Work incrementally and test each increment. Save often.

1. **Add** external class folder.
2. **Then**, implement class `FileCatalogLoader` from scratch. Use

   the `TestFileCatalogLoader` driver to test your implementation.

3. **Next**, implement the method `BrightFreshMilk.writeFile`.

4. **Finally**, compile the class `BrightFreshMilk`, and execute the

   class `BrightFreshMilk` by issuing the following command at the

   command prompt:

   ```
   C:\>java BrightFreshMilk catalog.dat
   ```

   Sales information has been hard-coded in the `BrightFreshMilk`

   template provided by iCarnegie.

   - If the user displays the catalog, the output should be:

     ```
     A001 milk,400ml 10

     A002 skim milk,800ml 18

     B001 solid,250ml 8.5

     B002 solid,400ml 14

     C001 chocolate flavor,270ml 5

     C002 strawberry flavor,400ml 10

     D001 taro flavor,250ml 13.1

     D002 apple flavor,400ml 16
     ```

- If the user saves the sales information in plain text, a file with the following content should be created:

```
------------------------
Order 1

5 C001 5

Total = 25
------------------------
Order 2

2 C002 10
2 A001 10

Total = 40
------------------------
Order 3

1 B002 14

Total = 14
```

- If the user saves the sales information in HTML, a file with the following content should be created:

```
<html>
  <body>
    <center><h2>Orders</h2></center>
    <hr>
    <h4>Total = 25</h4>
      <p>
        <b>code:</b> C001<br>
        <b>quantity:</b> 5<br>
        <b>price:</b> 5
      </p>
    <hr>
    <h4>Total = 40</h4>
      <p>
        <b>code:</b> C002<br>
        <b>quantity:</b> 2<br>
        <b>price:</b> 10
      </p>
      <p>
```

```
            <b>code:</b> A001<br>
            <b>quantity:</b> 2<br>
            <b>price:</b> 10
        </p>
    <hr>
    <h4>Total = 14</h4>
        <p>
            <b>code:</b> B002<br>
            <b>quantity:</b> 1<br>
            <b>price:</b> 14
        </p>
  </body>
</html>
```

- If the user saves the sales information in XML, a file with the following content should be created:

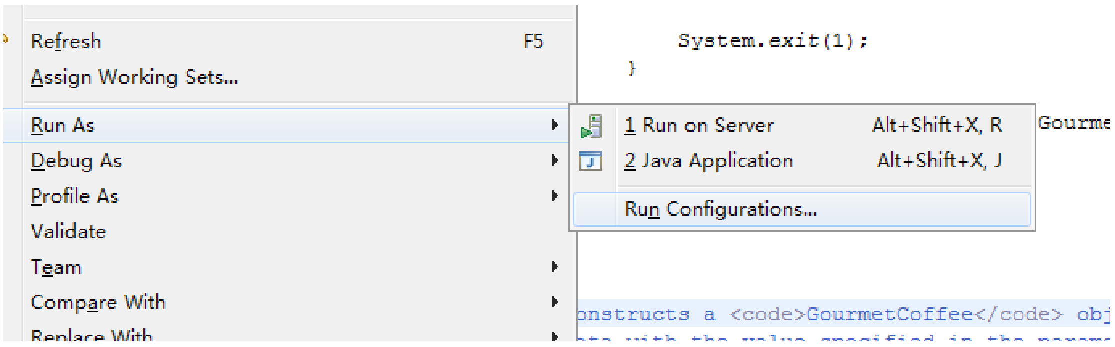
```
<Sales>
  <Order total="25">
    <OrderItem quantity="5" price="5">C001</OrderItem>
  </Order>
  <Order total="40">
    <OrderItem quantity="2" price="10">C002</OrderItem>
    <OrderItem quantity="2" price="10">A001</OrderItem>
  </Order>
  <Order total="14">
    <OrderItem quantity="1" price="14">B002</OrderItem>
  </Order>
</Sales>
```

Submission

Upon completion, submit **only** the following:

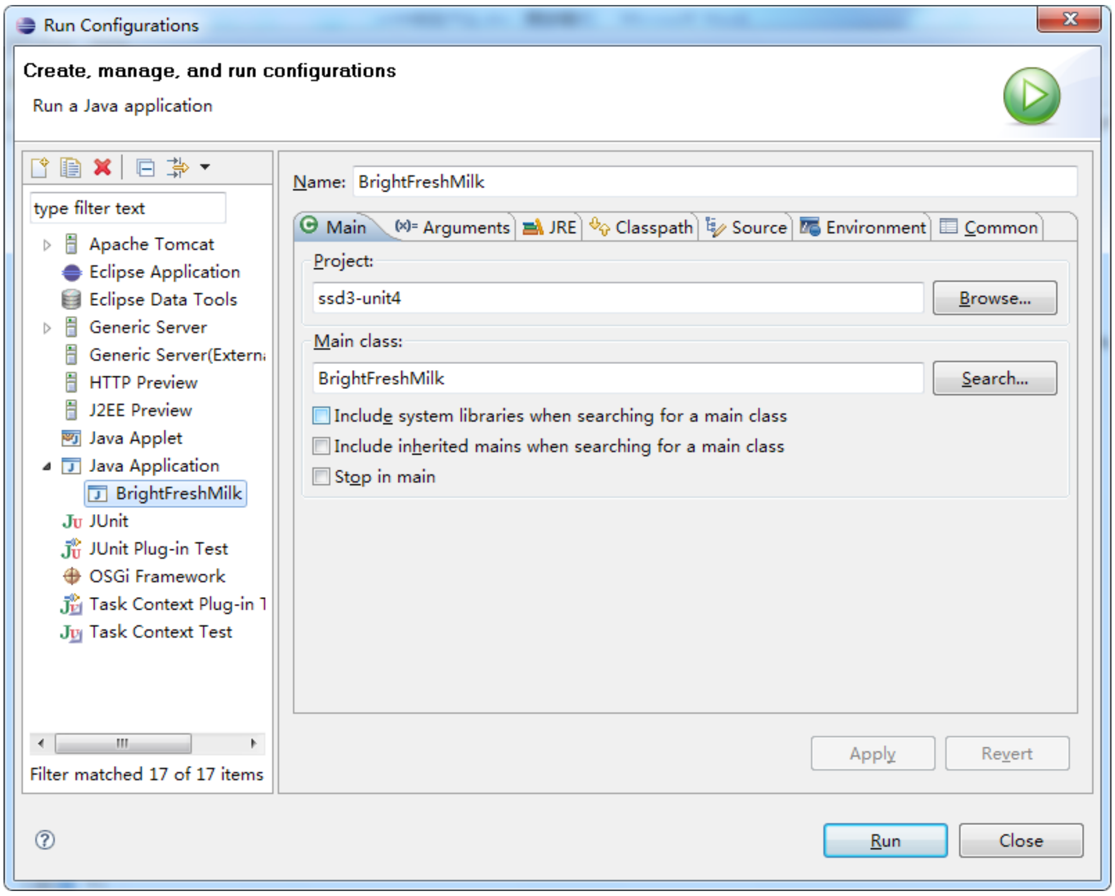
1. `FileCatalogLoader.java`
2. `BrightFreshMilk.java`

Eclipse 填写运行参数的方法



```
        System.exit(1);
    }
Gourme
onstructs a <code>GourmetCoffee</code> obj
```

鼠标双击 Java Application 下方出现 BrightFreshMilk，点击 Arguments 选项卡。

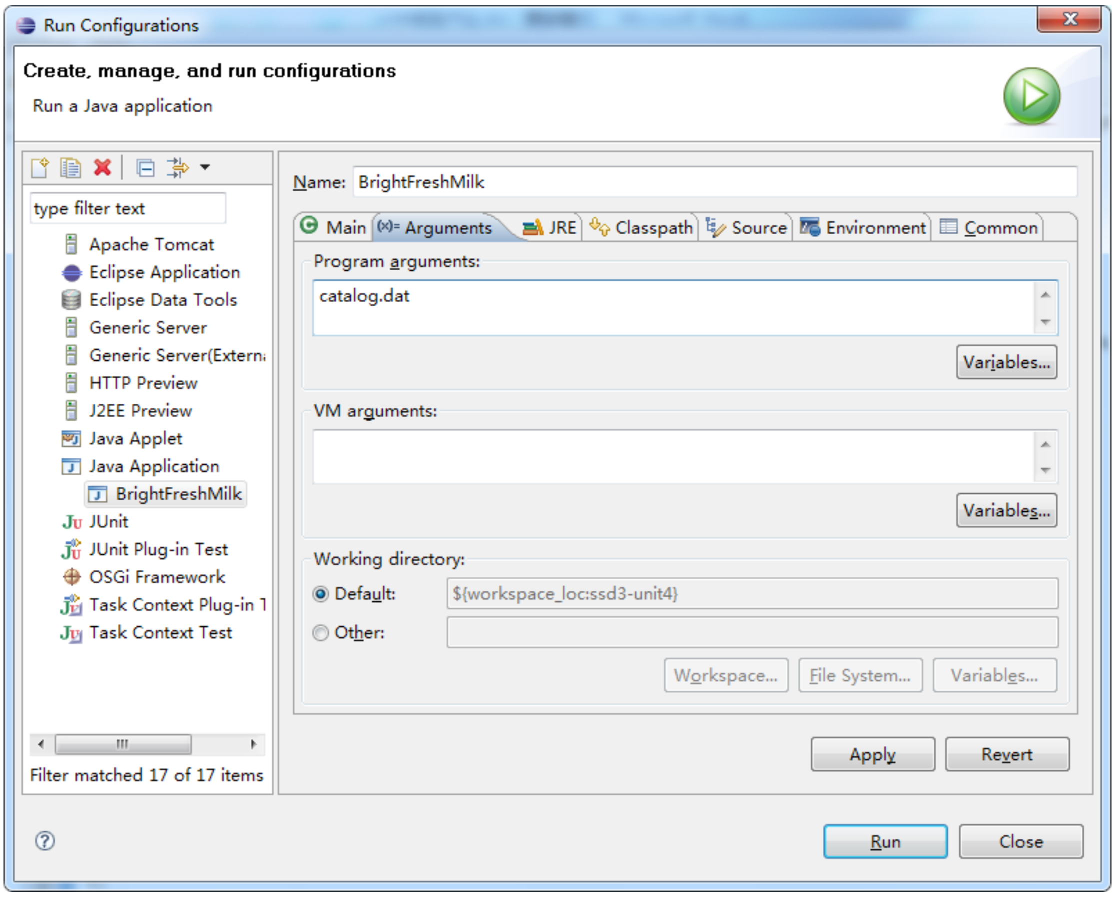


Name: BrightFreshMilk

Project: ssd3-unit4

Main class: BrightFreshMilk

- [ ] Include system libraries when searching for a main class
- [ ] Include inherited mains when searching for a main class
- [ ] Stop in main


Run Configurations

Create, manage, and run configurations

Run a Java application

type filter text

Apache Tomcat
Eclipse Application
Eclipse Data Tools
Generic Server
Generic Server(Extern
HTTP Preview
J2EE Preview
Java Applet
Java Application
BrightFreshMilk
JUnit
JUnit Plug-in Test
OSGi Framework
Task Context Plug-in T
Task Context Test

Filter matched 17 of 17 items

Name: BrightFreshMilk

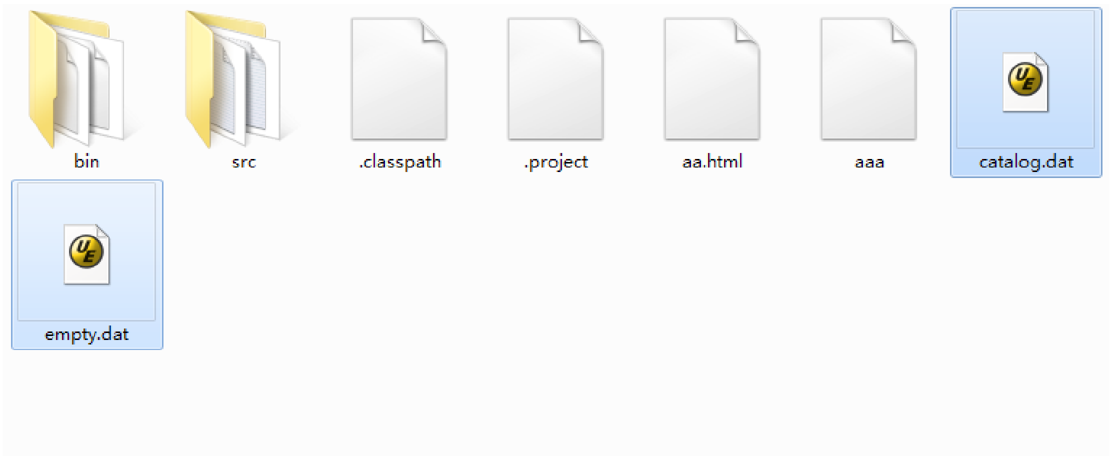
Main | Arguments | JRE | Classpath | Source | Environment | Common

Program arguments:
```
catalog.dat
```

VM arguments:

Working directory:
Default: ${workspace_loc:ssd3-unit4}
Other:

在 Program arguments 输入框内填写数据文件名称，该数据文件拷贝到工程项目目录文件下


bin
src
.classpath
.project
aa.html
aaa
catalog.dat
empty.dat