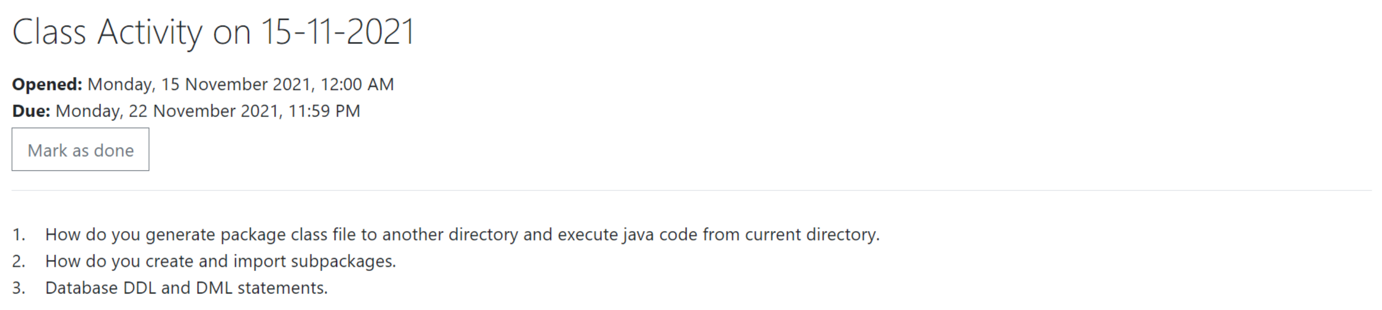
CSE1007 Java Programming Theory A2+TAA2 on 15th November 2021

Sharadindu Adhikari | 19BCE2105



Class Activity on 15-11-2021

Opened: Monday, 15 November 2021, 12:00 AM
Due: Monday, 22 November 2021, 11:59 PM

Mark as done

1. How do you generate package class file to another directory and execute java code from current directory.
2. How do you create and import subpackages.
3. Database DDL and DML statements.

1. How do you generate package class file to another directory and execute java code from current directory.

Let's say, we have:

- two java files named [**Java.java** and **C.java**] and two directories named [**E:\Javaprograms** and **E:\Cprograms**].
- The first java file **Java.java** is inside **E:\Javaprograms** directory and the second java file **C.java** is inside **E:\Cprograms** directory.

Now I've to execute **Java.class** file of **E:\Javaprograms** directory inside **E:\Cprograms** directory.

There are various steps to follow to **run java class file which is in other directory**,

1) In the first step, we are creating a java file named **Java.java** in **E:\Javaprograms** directory.

Java.java

```
class Java {
    public void display() {
        System.out.println("Java.java file is in E:\\Javaprograms directory");
    }
}
```

2) In the second step, we will compile **Java.java** file in **E:\Javaprograms** directory so we will perform a few steps.

i. Open command prompt or terminal from the start menu.
ii. After open terminal, we have to reach the path where our Java.java file has been stored.

```
C:\Users> cd\ and press enter
    [To move to the base directory]
C:\>  e: and press enter and then cd Javaprograms and again press enter.
    [To move to the directory where our Java.java file is stored.]
E:\Javaprograms> javac Java.java and press enter
    [If file is successfully compiled then class file will
    generate in the same directory E:\Javaprograms.]
```

3) In the third step, we will see what will happen if we run java class file named **Java.class** of [**E:\Javaprograms**] in another directory named [**E:\Cprograms**].

Here, we are creating another java file named **C.java** in **E:\Cprograms** directory.

C.java:

```
class C {
    public static void main(String[] args) {
        System.out.println("C.java file is in E:\Cprograms directory");
        // Here we are creating an object of Java.java class
        // of E:\Javaprograms
        Java ja = new Java();
        Ja.display();
    }
}
```

Note: If we compile the above program then we will get compile-time error class Java not found because this class is located in **E:\Javaprograms** directory so we try to execute **Java.class** inside **E:\Cprograms** then we will get an error so to overcome this problem when we include **Java.class** file of **E:\Javaprograms** directory in this **E:\Cprograms** directory.

4) In the fourth step, we will see how to include **Java.class** file of **E:\Javaprograms** in this **E:\Cprograms** directory. With the help of **–cp** option we can include **Java.class** of **E:\Javaprograms** in this **E:\Cprograms** directory.

Syntax for Compiling:

```
    E:\Cprograms> javac -cp E:\Javaprograms C.java
```

–cp E:\Javaprograms: **-cp** with pathname (we will provide the path of the included file and here included file is **Java.class** is in **E:\Javaprograms** directory).

C.java: This is the name of the compiled class.

Syntax for Executing:

```
    E:\Cprograms> java -cp E:\Javaprograms; C
```

5) In the fifth or final step, we will understand with the help of Example,

Example:

Java.java inside E:\Javaprograms

```
class Java {
    public void display() {
        System.out.println("Java.java file is executing in different directory");
    }
}
```

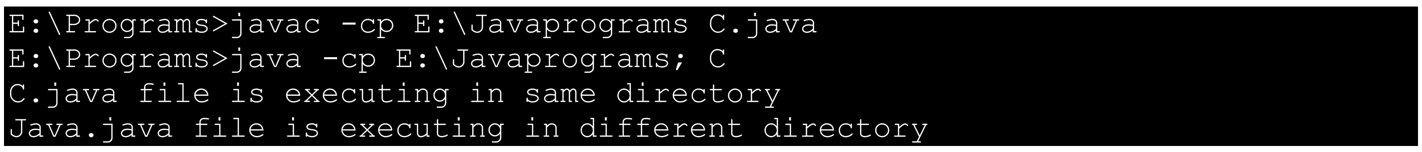
C.java inside E:\Cprograms

```
class C {
    System.out.println("C.java file is executing in same directory");
    public static void main(String[] args) {
        // Here we are creating an object of Java.java class
        // of E:\Javaprograms
        Java ja = new Java();
        ja.display();
    }
}
```

We will compile and execute C class of E:\Cprograms directory and in that we are executing another java class named Java.class of E:\Javaprograms inside E:\Cprograms with the help –cp option.

```
    E:\Cprograms> javac -cp E:\Javaprograms C.java
    E:\Cprograms> java -cp E:\Javaprograms; C
```

Output:

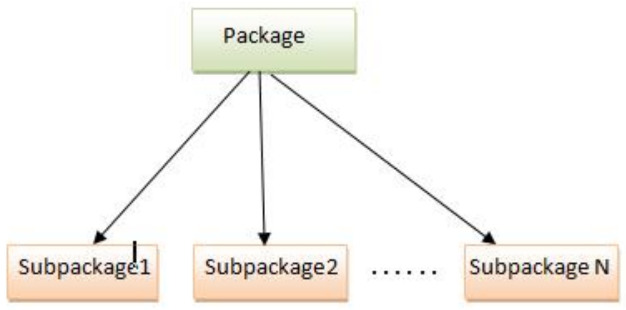


```
E:\Programs>javac -cp E:\Javaprograms C.java
E:\Programs>java -cp E:\Javaprograms; C
C.java file is executing in same directory
Java.java file is executing in different directory
```

2. How do you create and import subpackages.

Creating subpackages:

Subpackages are created to categorize/divide a package further. It's similar as creating sub folder inside a folder to categorize it further so that we can organize our content much better, which will make easy to access the content. A package may have many sub packages inside it.



For an example in our computer generally we create directory like songs to store songs inside it. Then inside that we may create sub directories like hindi songs and english songs or old songs and new songs to categories the songs directory further. Doing this help us to organize or access the songs easily. The same thing applies with sub packages as well.

```
package mypack.testpack;

 class MySubPackageProgram {
   public static void main(String args []) {
      System.out.println("My sub package program");
    }
 }
```

We can see here the package name is `mypack.testpack` which is a subpackage. Let's compile and run this program by executing the following commands:

```
javac mypack\testpack\MySubPackageProgram.java
java mypack.testpack.MySubPackageProgram
```

Output:

```
My sub package program
```

Importing subpackages:

To access the classes or interfaces of a sub package, we need to import the sub package in our program first. Importing the parent package of a sub package doesn't import the sub packages classes or interfaces in your program. They must be imported explicitly in our program to access their classes and interfaces.

For example, importing package mypack in your program will not import the classes of sub package testpack given above. Importing sub packages is same as importing packages. The syntax of importing a sub package is :

```
  // To import all classes of a sub package
 import packagename.subpackagename.*;
  // To import specific class of a sub package
 import packagename.subpackagename.classname;

 Example:

 import mypack.testpack.*;
 import mypack.testpack.MySubPackageProgram;
```

3. Database DDL and DMS Statements.

DDL:

DDL allows us to create SQL statements to make operations with database data structures (schemas, tables etc.).

These are SQL DDL commands list and examples:

1. CREATE

CREATE statement is used to create a new database, table, index or stored procedure.
Create database example:

```
CREATE DATABASE explainjava;
```

Create table example:

```
CREATE TABLE user (
  id INT(16) PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL
);
```

2. DROP

DROP statement allows you to remove database, table, index or stored procedure.
Drop database example:

```
DROP DATABASE explainjava;
```

Drop table example:

```
DROP TABLE user;
```

3. ALTER

ALTER is used to modify existing database data structures (database, table).
Alter table example:

```
ALTER TABLE user ADD COLUMN lastname VARCHAR(255) NOT NULL;
```

4. RENAME

RENAME command is used to rename SQL table.
Rename table example:

```
RENAME TABLE user TO student;
```

5. TRUNCATE

TRUNCATE operation is used to delete all table records. Logically it's the same as DELETE command.

Differences between DELETE and TRUNCATE commands are:

- TRUNCATE is really faster
- TRUNCATE cannot be rolled back
- TRUNCATE command does not invoke ON DELETE triggers

Example:

```
TRUNCATE student;
```

DML:

DML is a Data Manipulation Language, it's used to build SQL queries to manipulate (select, insert, update, delete etc.) data in the database.

This is DML commands list with examples:

1. SELECT

SELECT query is used to retrieve a data from SQL tables.

Example:

```
SELECT * FROM student;
```

2. INSERT

INSERT command is used to add new rows into the database table.

Example:

```
INSERT INTO student (name, lastname) VALUES ('Dmytro', 'Shvechikov');
```

3. UPDATE

UPDATE statement modifies records into the table.

Example:

```
UPDATE student SET name = 'Dima' WHERE lastname = 'Shvechikov';
```

4. DELETE

DELETE query removes entries from the table.

Example:

```
DELETE FROM student WHERE name = 'Dima';
```