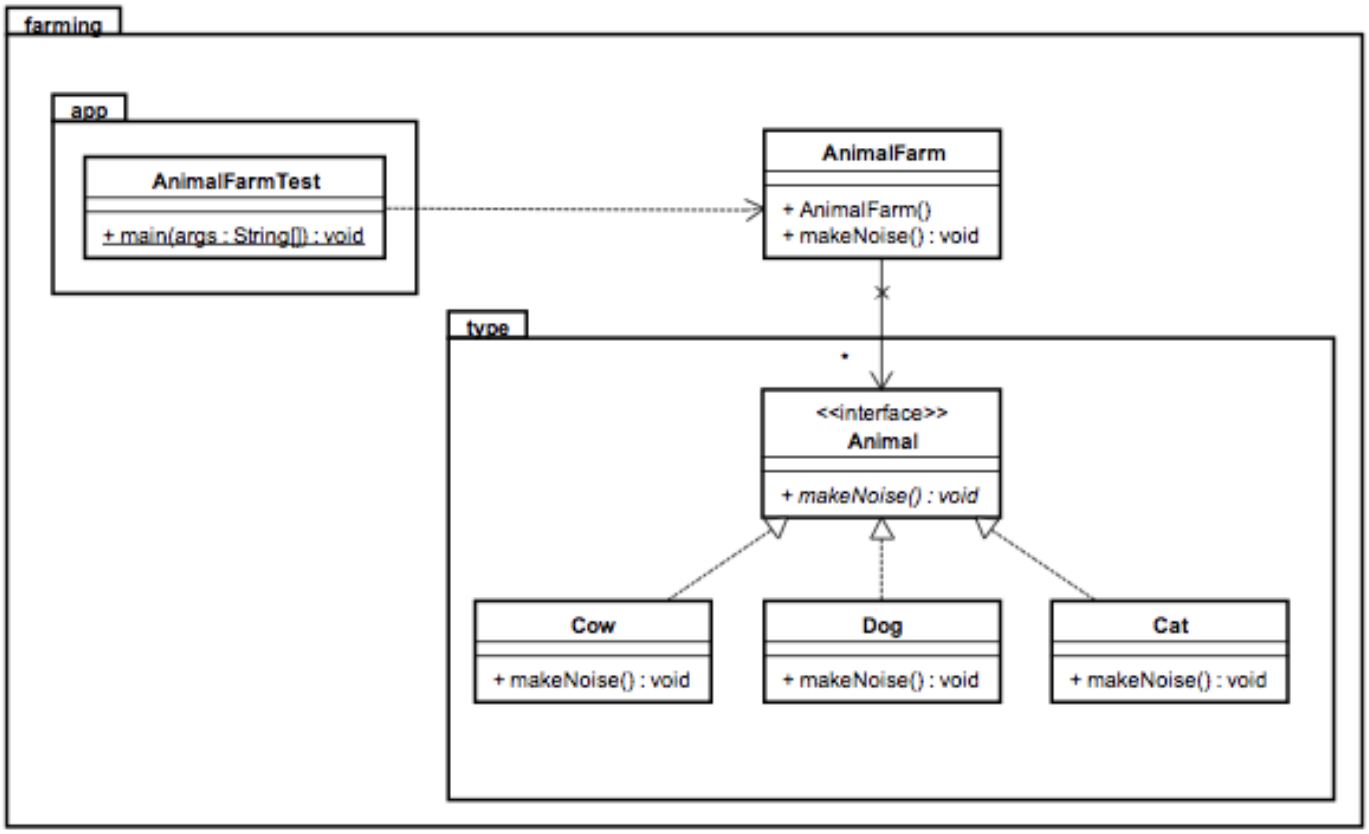
Classpath

Resource: http://blue.smu.edu.sg/classpath-resource-V3.zip

1. **[Difficulty: **]** Given the following class diagram:



   a. Package the Java classes accordingly.

   b. Place your Java classes in the proper sub-directories in the `src` folder.

c. Write a one-liner command in compile.bat to compile `AnimalFarmTest` so that all Java files are compiled and the class files are placed in the `classes` folder.

d. Write a one-liner command in run.bat to run class `AnimalFarmTest`.

2. **[Difficulty: **]** You are now given the following:

```
Q2\
+- compile.bat
+- run.bat
+- lib\
    +- farming.jar   //contains Animal.class, Cat.class, Dog.class & Cow.class
+- classFiles\
+- sourceFiles\
```

a. Place your AnimalFarm.java and AnimalFarmTest.java in the proper sub-directories in the `sourceFiles` folder. Follow the class diagram given for Question 1

b. Write a one-liner command in compile.bat to compile `AnimalFarmTest` so that all Java files are compiled and the class files are placed in the `classFiles` folder.

c. Write a one-liner command in run.bat to run class `AnimalFarmTest`.

3. **[Difficulty: **]** You are now given the following:

```
Q3\
+- compile.bat
+- run.bat
+- lib\
    +- types.jar // contains Animal.class, Cat.class, Dog.class, Cow.class
    +- farm.jar  // contains AnimalFarm.class
+- classFiles\
+- sourceFiles\
```

a. Place your AnimalFarmTest.java in the proper sub-directories in the `sourceFiles` folder.

b. Write a one-liner command in compile.bat to compile `AnimalFarmTest` so that all Java files are compiled and the class files are placed in the `classFiles` folder.
Question: Is `types.jar` required on the classpath?

c. Write a one-liner command in run.bat to run class `AnimalFarmTest`.

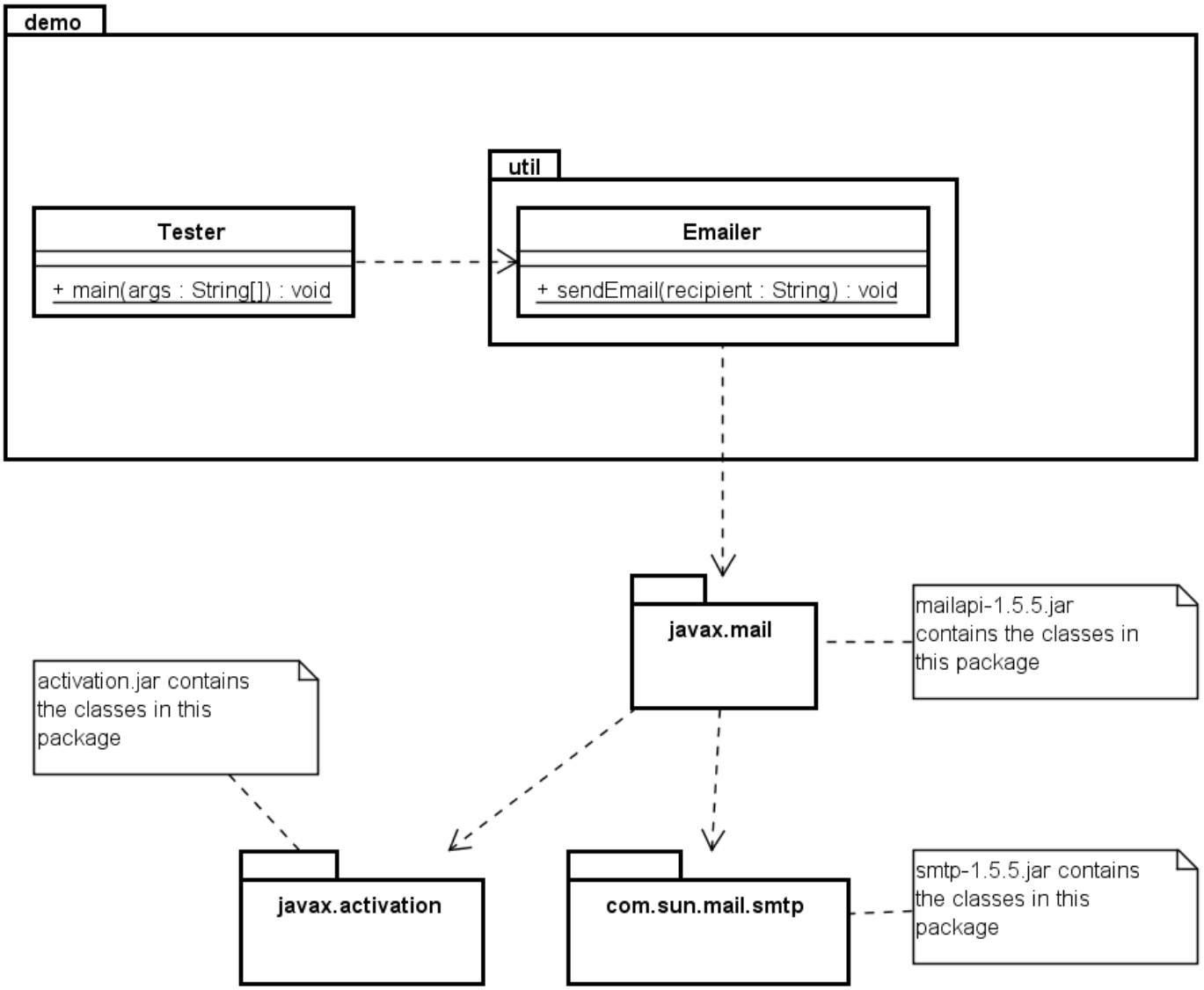
4. **[Difficulty: **]** Look at the directory structure shown below and the files given to you in the resources:

```
D:\ooad\
  +- my_code\
  |  +- smu\
  |     +- bidding\
  |     |  +- App.java
  |     |
  |     +- object\
  |        +- Venue.java
  |
  +- my_compiled\
  |
  +- ben_code\
  |  +- smu\
  |     +- dm\
  |     |  +- CourseManager.class
  |     |  +- StudentManager.class
  |     |
  |     +- object\
  |        +- Course.class
  |        +- Student.class
  |
  +- download\
  |  +- apache\
  |  |  +- com\
  |  |     +- opencsv\
  |  |        +- CSVReader.class
  |
  +- compile.bat
  +- run.bat
```

a. Create `compile.bat` such that all source files are compiled and the generated class files are placed in the folder "`my_compiled`".

b. Create run.bat to run class `App`.

5. **[Difficulty: **]** Given the following class diagram:



a. Package the Java classes accordingly.

b. Place your Java classes in the proper sub-directories in the `source` folder.

c. Write a one-liner command in compile.bat to compile `Tester` so that all Java files are compiled and the class files are placed in the `out` folder. How many of the jar files do you need during compilation?

d. Write a one-liner command in run.bat to run class `Tester`.

Task:

1. Note that smtp.smu.edu.sg is only available when you are on SMU network. If you wish to utilize Google's SMTP server, what modifications do you need?

6. **[Difficulty: ***]** A JAR (**J**ava **AR**chive) is a way of packaging together all the resources associated with a program (class files, images, sounds, etc.). Putting your program in a JAR allows it to be distributed as a single executable file, saving space and simplifying the download process.

    a. Compile your java code, generating all the program's class files.

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

    b. Create a manifest file, `hello.mf` containing the following 2 lines:

```
Manifest-Version: 1.0
Main-Class: Hello
```

    Note: The name of the file should end with the `.mf` suffix. It is important that the file ends with a blank line.

    c. To create the JAR, type the following command:

```
jar cmf hello.mf hello.jar Hello.class
```

    Note:

    The command is

    jar cmf *manifest-file jar-file input-files*

    The *input-files* include any class files, images, sounds, etc. that your program uses. To find out more, type `"jar --help"` (without quotes) on your command prompt window.

    d. To execute your application, type the following command:

```
java -jar hello.jar
```

Task:

1. Download the Java file from https://docs.oracle.com/javase/tutorial/uiswing/examples/components/TableDemoProject/src/components/TableDemo.java (It is the first result when you google "TableDemo.java"). Package it into an executable jar without any modification to the Java file.

Note:: Since the application is GUI-based, you can also launch it by double-clicking the JAR file (only for windows users).

7. **[Difficulty: ***]** When you are using ArrayList, you can try "act cute" by using the "kawaii" sign. You are given the following class:

```
1   import java.util.ArrayList;
2
3   public class Q7 {
4      public static void main(String[] args) {
5          ArrayList container = new ArrayList();
5          container.add("apple");
6          container.add(1);
7          container.add(3.14);
8
9          for (Object c : container) {
10             System.out.println(c);
11         }
12     }
13  }
```

a. Do you observe any warning message when you try to compile the above Java file?

b. Now, try replacing line 5 with:

```
ArrayList<String> container = new ArrayList<>();
```

Is the compilation successful?

c. Now, try to compile the given `Utility.java` source code. Does the code compile? If no, what is the error message? Which of the following method invocations will work?

i. `doSomething(new List<Cat>());`
ii. `doSomething(new List<Animal>());`

d. Change the method signature of `doSomething` method in `Utility.java` to the following:

```
public static void doSomething(List<? extends Animal> aList) {
```

Check if your code compiles now?

e. Now, uncomment the following line before trying to compile again. What's your observation(s)? Which of the following method signatures of doSomething will work?

i. `public static void doSomething(List<? extends Animal> aList) {`
ii. `public static void doSomething(List<Animal> aList) {`

```
// aList.add(new Dog("Browny"));
```

8. **[Difficulty: ***]** Given `GenericExample.java`.

```
1   public class Q8 {
2      private String t;
3
4      public void set(String t) { this.t = t; }
5      public String get() { return t; }
6
7
8      public static void main(String[] args) {
9          Q8 example = new Q8();
10         example.set("Ah Seng");
11
12         String value = example.get();
13         System.out.println(value);
14
15     }
16  }
```

a. Compile it. (No issues. The code is perfect! :)

b. Now, try modifying line 9 to the following given code. Does it compile now?

```
Q8<String> example = new Q8<String>();
```

c. Modify the code to include the following, and try compiling again. Does it compile now? Will It work it if you change it to

```
Q8<Integer> example = new Q8<Integer>();
```

Read the references below to understand more about generics.

```
public class Q8<T> {
   private T t;

   public void set(T t) { this.t = t; }
   public T get() { return t; }

   // main method excluded
}
```

Reference:

1. https://docs.oracle.com/javase/tutorial/extra/generics/index.html
2. https://www.baeldung.com/java-generics-interview-questions
3. http://www.angelikalanger.com/GenericsFAQ/JavaGenericsFAQ.html

9. **[Difficulty: ***]** Java 8 introduces the "Default Method" (aka Defender method or virtual extension method) feature, which allows the developer to add new methods to the interfaces without breaking their existing implementation. It provides the flexibility to allow interfaces to define implementation which will be used as the default in a situation where a concrete class fails to provide an implementation for that method.

   To implement a default method, the default keyword is used. A Human Interface is given below:

```
public interface Human {
   public default void smoke() {
       System.out.println("This is rubbish!");
   }
}
```

   a. Open Student.java and note that we did not implement the smoke() method in the concrete subclass. Compile and execute Student. Note the output.

   b. Open Smoker.java. This class implements both the Human and Flammable interfaces that both have a default smoke() method. Are you able to compile Smoker class? What is the error message?

   c. Override the smoke method in Smoker class. We want to have the default behavior of both Human and Flammable interface. Insert the given code. Compile and execute the DefaultTest class.

```
public class Smoker implements  Human, Flammable {

   @Override
   public void smoke() {
       Human.super.smoke(); // note: we need to prefix it with the interface name
       Flammable.super.smoke();
   }
}
```

Reference:

1. http://cr.openjdk.java.net/~briangoetz/lambda/Defender%20Methods%20v4.pdf
2. https://docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html

- END -