CMSC131

Command line Java

Command Line Java

- Assume we have a Java class named **`Classname`** which contains a static main method.

- Compiling basic idea:

  `javac  Classname.java`

- This creates a bytecode file **`Classname.class`**

- Running basic idea:

  `java  Classname`

NOTE: If **`Classname`** uses other classes it will look for the **`.class`** files for those, and if it doesn't find them it will attempt to compile the **`.java`** files of those classes as well.

System Requirements

- Have the JDK installed.
- The command search path (**`PATH`**) must include the directory in which the **`java`** and **`javac`** executables are located.
- The class search path (**`CLASSPATH`**) must include the directories in which your Java source code and **`.class`** files are located.

Command Line Arguments

Recall the main method appears as:

```
public static void main (String[] args)
```

If run from the command line, any command line arguments are placed into the **`args`** array.

Try it on **terpconnect.umd.edu**

- To log into one of these machines, you might need to activate your TerpConnect account (which can be done at **http://www.it.umd.edu/new/**). You can try to log into one of these machines by using a ssh client. NOTE: Your password will not appear as you type it.
- Mac OS X provides one called Terminal. After you open Terminal, type the following to connect:

```
ssh  YOUR_USER_NAME@terpconnect.umd.edu
```

where you replace YOUR_USER_NAME with your actual UMD login name. When it asks for a password, use your UMD password.

- For Windows you can download one called PuTTy at **http://tinyurl.com/1buv** one and then when you run that give the host name as **terpconnect.umd.edu** and use your UMD login name and password.

Compile and run on **terpconnect**

- After you log into the machine, you can type the following to get a copy of the file into your directory:

  `cp /pub/egolub/ExampleOne.java .`
- To compile it type: **`javac ExampleOne.java`**
- To run it type: **`java ExampleOne`**

- After doing this, run it again by typing the following instead: **`java ExampleOne hello there`**

- To see the program, type: **`more Example.java`**
- To see that a class file was generated for Example.java, type: **`ls`**

Compile and run on **terpconnect**

The next example will use several Java source code files that are part of a package, so you'll need to make a folder, go into that folder, copy all of the files into that folder, and then to compile them.

```
mkdir   ~/ExamplePackage

cd   ~/ExamplePackage

cp   /pub/egolub/ExamplePackage/ExampleTwo.java   .
cp   /pub/egolub/ExamplePackage/Animal.java   .
cp   /pub/egolub/ExamplePackage/Cat.java   .
```

Compile and run on **terpconnect**

- **ExampleTwo** uses a **Cat** which implements **Animal**.
- To compile **ExampleTwo** as well as everything it uses, you'll need to do things a little differently since they are in a package. One way which will work is to return to the home directory:

  ```
  cd   ~
  ```

  and then give the command to compile:

  ```
  javac   ExamplePackage/ExampleTwo.java
  ```

- To run it type: `java ExamplePackage/ExampleTwo`

- To see that class files were created for **Animal** and **Cat** as well as **ExampleTwo**, type: `ls ExamplePackage/`