Running JVM Applications on the Command Line

1. Setting Up the Project

Assume the following project structure for ByteCraft Pro:

```
ByteCraftPro/
|
├── src/
|   ├── com/byteminds/bytecraftpro/main/Application.java
|   ├── com/byteminds/bytecraftpro/util/Logger.java
|   └── ...
├── bin/ (compiled `.class` files will be placed here)
└── lib/ (external libraries, e.g., `.jar` files)
```

2. Compilation and Execution Steps

Step 1: Compile the Java Files

Navigate to the project root directory (`ByteCraftPro`) in the terminal and run:

```
javac -d bin -cp lib/* src/com/byteminds/bytecraftpro/main/Application.java
```

Explanation:

- `-d bin`: Specifies the output directory for `.class` files.
- `-cp lib/*`: Adds all libraries in the `lib` directory to the classpath.
- `src/com/byteminds/bytecraftpro/main/Application.java`: The main Java file to compile.

This will generate `.class` files in the `bin` directory, preserving the package structure.

Step 2: Run the Application

Run the compiled application using:

```
java -cp bin:lib/* com.byteminds.bytecraftpro.main.Application
```

Explanation:

- `-cp bin:lib/*`: Specifies the classpath, including the `bin` directory and external libraries. Use `;` instead of `:` on Windows.
- `com.byteminds.bytecraftpro.main.Application`: Fully qualified name of the main class containing the `public static void main(String[] args)` method.

3. Handling System Properties and Command-Line Arguments

System Properties

You can pass system properties using the `-D` option.

Example:

```
java -Dconfig.path=/path/to/config -cp bin:lib/* com.byteminds.bytecraftpro.main.Application
```

Access the property in your code using:

```
String configPath = System.getProperty("config.path");
```

Command-Line Arguments

Command-line arguments are passed after the class name.

Example:

```
java -cp bin:lib/* com.byteminds.bytecraftpro.main.Application arg1 arg2
```

Access the arguments in the `main` method:

```
public static void main(String[] args) {
    for (String arg : args) {
        System.out.println("Argument: " + arg);
    }
}
```

4. Practical Considerations

Classpath Considerations

- Use `-cp` to include multiple `.jar` files and directories.
- Ensure libraries are placed in the `lib` directory for consistency.

Debugging Logs

Use a utility class like `Logger` to standardize debug messages during execution.

5. Example Run

Assuming the Application class uses a system property (config.path) and reads command-line arguments:

```
java -Dconfig.path=./config -cp bin:lib/* com.byteminds.bytecraftpro.main.Application user1 admin
```

Output:

```
Config Path: ./config
Argument: user1
Argument: admin
```