Java Programming Cheat Sheet PAGE 1 of 2 BY SETH KENLON

Java runs in a Java Virtual Machine (JVM), a layer that translates Java code into bytecode compatible with your operating system. Get an open source JVM from openjdk.java.net or developers.redhat.com/products/openjdk/download

Java Packages

Related classes are grouped into a package. Declare a package name at the top of your code.

```
package com.opensource.hello;
/* @author your-name */
```

When creating libraries, you can create packages within a package to provide access to each unique class.

```
package com.opensourc.greeting;
/* @author your-name */
```

Java Imports

When importing libraries needed for your Java code, use the `import` key word. Imports work based on your environment's Java path, and open source libraries can be bundled with your application (license permitting.)

```
package com.opensource.hello;
/* @author your-name */

import java.lang.System;
import javax.swing.*;

//code...
```

Java Variables

Java is strongly typed, meaning all variable types must be declared when created. *Local* variables may be created inside a method, *instance* variables may be created inside a class, and *static* variables may be shared across instances.

```
//code...
public class Greeting {
  static int num = 42;
  public static String namer() {
    String name = "Java";
    int number =1;
    double bignum = 3.1415926535897932384;
    float smallnum = 3.141592;
    char character = 'a';
    boolean toggle = true;
    return name; }
}
```

Java Classes

A class file must contain one `public` class. Other classes may exist elsewhere and may not be public. By convention, class names start with a capital letter.

```
package com.opensource.hello;
/* @author your-name */

import java.lang.System;

public class Greeting {
  public static String namer() {
  //code...
  }
}
```

Java Methods

Java methods may be `public` (accessed by any other class), `private` (known only within a class), `protected` (unavailable to an unrelated class), or default. Provide the type of returned data, such as `void`, `int`, `float`, etc.

```
//code...
public static String namer() {
    String name = "Java";
    return name; }
  }

public static void main(String[] args) {
  String player = greeting.namer();
  System.out.printf("%s,%s", named, num);
  }
}
```

Try and Catch

To catch errors in Java, start with `try`, fall back on `catch`, and end with `finally`. Should the `try` clause fail, then `catch` is invoked, and in the end, there's `finally` to perform some action regardless of the results.

```
try {
        cmd = parser.parse(opt, args);

        if(cmd.hasOption("help")) {
                HelpFormatter helper = new HelpFormatter();
                helper.printHelp("Hello <options>", opt);
                System.exit(0);
                }
        else {
                if(cmd.hasOption("shell") || cmd.hasOption("s")) {
                String target = cmd.getOptionValue("tgt");
                } // else
        } // if
} catch (ParseException err) {
        System.out.println(err);
        System.exit(1);
        } //catch
        finally {
                new Hello().helloWorld(opt);
        } //finally
} //try
```

Arguments

Passing arguments into a Java application is done with the `args` keyword, preceded by the type of acceptable arguments (such as `String`, `int`, and so on).

```
//code...
public static void main (String[] args) {
  System.out.printf("You rolled a ");
  DiceRoller App =
    new DiceRoller(
    Integer.parseInt(args[0]) );

  App.Roller();
 }
}
```

Run a .java file

Java files, usually ending in `.java`, can be run from your IDE or in a terminal using the `java` command. If an application is complex, however, running a single file may not be useful. Add arguments as appropriate.

```
$ java ./Example.java
$ java ./diceroller.java 20
You rolled a 17
```

Run a JAR file

Usually, Java applications are distributed as Java Archives (JAR) files, ending in `.jar`. To run a JAR file, double-click its icon or launch it from a terminal.

```
$ java -jar /path/to/Example.jar
```