Programming Cheatsheet v2.0

1. Getting Started with Java

```
// run in cmd.exe / terminal
// compile
javac -d bin -cp bin;lib -sourcepath src src/com/example/App.java
// run
java -cp bin;lib com.example.App
// run with JavaFX modules
java --module-path "C:\javafx-sdk-21\lib" --add-modules javafx.controls,javafx.fxml -cp bin;lib com.example.App
```

Developing a Java program

Program 1.1.1 Using a command line argument

```
public class HowAreYou
{
    public static void main(String[] args)
    {
        System.out.print("Hi, ");
        System.out.print(args[0]);
        System.out.print(", How are you?");
    }
}
```

The program shows the way in which we can convey the action of our program to the user by printing a message to the screen. The program takes the name of the user from the command line and uses it in the message.

```
> java HowAreYou Jon
Hi, Jon, How are you?
> java HowAreYou Ann
Hi, Ann, How are you?
```

2. Print Formatting

```
System.out.printf("%d %s", 42, "Hello");
                  ^^^^^^^^^^^^^^^^^^^^^
                  format string
                  ^^^^^^^^^^^^^^^^^^^^^
                  conversion
                  ^^^^^^^^^^^^^^^^^^^^^
                  arguments
```

Anatomy of a formatted print statement

conv	char	byte	short	int	long	scientific	decimal	hexadecimal	octal	boolean	char	Character	Byte	Short	Integer	Long	BigInteger	float	double	BigDecimal	Number	Object
%c	c	120	120	120	120	120	120	120	170	true	c	c	120	120	120	120	120	120.0	120.0	120.0	120	120
%d	d	120	120	120	120	120	120	120	170	true	120	120	120	120	120	120	120	120.0	120.0	120.0	120	120
%o	o	120	120	120	120	120	120	120	170	true	170	170	170	170	170	170	170	170.0	170.0	170.0	170	170
%x	x	120	120	120	120	120	120	120	170	true	78	78	78	78	78	78	78	78.0	78.0	78.0	78	78
%f	f	120.000000	120.000	120.000	120.000	120.000	120.000	120.000	170.000	true	120.000	120.000	120.000	120.000	120.000	120.000	120.000	120.000	120.000	120.000	120.000	120.000
%e	e	1.200000e+02	1.200e+02	1.200e+02	1.200e+02	1.200e+02	1.200e+02	1.200e+02	1.700e+02	true	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02
%s	s	120	120	120	120	120	120	120	170	true	"x"	"x"	"120"	"120"	"120"	"120"	"120"	"120.0"	"120.0"	"120.0"	"120"	"120"
%b	b	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true	true

Format conversions for primitives and Object (see the Formatter for many other options)

3. Standard Input Methods

```
public class Hello
{
    public static void main(String[] args)
    {
        int    iVar = Integer.parseInt(System.in.nextLine());
        long   lVar = Long.parseLong(System.in.nextLine());
        double dVar = Double.parseDouble(System.in.nextLine());
        boolean bVar = Boolean.parseBoolean(System.in.nextLine());
        String cVar = System.in.nextLine().charAt(0);
        String sVar = System.in.nextLine();
        System.out.printf("%d %d %d %d %c %s", iVar, lVar, dVar, bVar, cVar, sVar);
    }
}
```

Program 1.1.2 Interactive user input

```
public class HowAreYouInteractive
{
    public static void main(String[] args)
    {
        System.out.print("What is your name? ");
        String name = System.in.nextLine();
        System.out.print("How old are you? ");
        int age = Integer.parseInt(System.in.nextLine());
        System.out.print("Are you married? ");
        boolean married = Boolean.parseBoolean(System.in.nextLine());
        System.out.printf("Hi %s, %d years old, married: %b", name, age, married);
    }
}
```

The program asks a simple question-prompt, the type-variable, and of which it is a subset of the type-variable, and the variable-name, and then it reads the input from the keyboard and stores it in the variable. Then we display back to the user what we have. We do questions. We ask the user a question, we read the answer, we store it in a variable, and we use the variable in a sentence. We do this over and over. We do questions. We do questions. We do questions. We do questions. We do questions. We do questions.

```
> java HowAreYouInteractive
What is your name? Jonathan Cazalet
How old are you? 42
Are you married? true
Hi Jonathan Cazalet, 42 years old, married: true
```

4. Creating Methods

```
modifiers   type   method-name(parameters-list)
public static void main(String[] args)
{
    System.out.print("Hi");
    System.out.print(" Jon");
    System.out.print(" How are you?");
}
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
method-signature
```

Anatomy of a static method

```
modifiers   type   method-name(parameters-list)
public static void main(String[] args)
{
    System.out.print("Hi");
    System.out.print(" Jon");
    System.out.print(" How are you?");
}
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
method-signature
```

Anatomy of a static method

4.1. Method Overloading

Overloading - Same methods with different signatures are different static methods. For example, we often want to be able to add two numbers together. We can add two integers, we can add two floating-point numbers, we can add two strings together, we can add two matrices together, we can add two vectors together, we can add two complex numbers together, we can add two fractions together, we can add two dates together, we can add two times together, we can add two currencies together, we can add two money together, we can add two weights together, we can add two distances together, we can add two speeds together, we can add two accelerations together, we can add two forces together, we can add two pressures together, we can add two temperatures together, we can add two ...

```
public static int add(int a, int b)
{
    return a + b;
}

public static double add(double a, double b)
{
    return a + b;
}
```

Anatomy of a method

4.2 Method with Multiple Return Statements (Prime Numbers Example)

Multiple return statements - You can put return statements in a method where ever you want. The method will stop at the first return statement it reaches, and the value in the return statement is reached. This primarily using function in a example of a function that is natural to define using multiple return statements:

```
public static boolean isPrime(int n)
{
    if (n <= 1) return false;
    if (n == 2) return true;
    if (n % 2 == 0) return false;
    return true;
}

public static int[] getPrimeFactors(int n)
{
    int[] primes = new int[0];
    for (int i = 2; i <= n; i++)
    {
        while (n % i == 0)
        {
            primes = append(primes, i);
            n /= i;
        }
    }
    return primes;
}
```

Anatomy of a static method

4.2.1 Factorial Recursion Method Example

```
public static long Factorial(int n)
{
    if (n <= 1) return 1;
    return n * Factorial(n-1);
}
```

5. Typical Method Examples

```
absolute value of a
int value
public static int abs(int n)
{
    if (n < 0) return -n;
    else       return n;
}

absolute value of a
double value
public static double abs(double n)
{
    if (n < 0) return -n;
    else       return n;
}

primality test
public static boolean isPrime(int n)
{
    if (n <= 1) return false;
    if (n == 2) return true;
    if (n % 2 == 0) return false;
    return true;
}

hypotenuse of
a right triangle
public static double hypotenuse(double a, double b)
{
    return Math.sqrt(a*a + b*b);
}

harmonic number
public static double Harmonic(int n)
{
    double sum = 0.0;
    for (int i = 1; i <= n; i++)
        sum += 1.0 / i;
    return sum;
}

uniform random
integer in [0,n)
public static int uniformInt(int n)
{
    return (int) (Math.random() * n);
}

draw a triangle
public static void drawTriangle(double x0, double y0,
                                double x1, double y1,
                                double x2, double y2)
{
    StdDraw.line(x0, y0, x1, y1);
    StdDraw.line(x1, y1, x2, y2);
    StdDraw.line(x2, y2, x0, y0);
}
```

Typical code for implementing functions (static methods)

5.1 Data Analysis Methods

```
public class StdStats
{
    double mean(double[] x);
    double var(double[] x);
    double stddev(double[] x);
    double min(double[] x);
    double max(double[] x);
    double[] cumulative(double[] x);
    void plot(PointsXY[] pts);
    void plot(LinesXY[] lines);
    void plotPoints(double[] y);
    void plotLines(double[] y);
    void plotBars(double[] y);
    void setXscale(double[] x);
    void setYscale(double[] y);
}
```

Some functions in StdStats for data analysis

5.2 Array Processing Code

```
create an array
with random values
double[] a = new double[N];
for (int i = 0; i < N; i++)
    a[i] = Math.random();

print the array values
one per line
for (int i = 0; i < a.length; i++)
    StdOut.println(a[i]);

find the maximum of
the array values
double max = a[0];
for (int i = 1; i < a.length; i++)
    if (a[i] > max) max = a[i];

compute the average
the array values
double sum = 0.0;
for (int i = 0; i < a.length; i++)
    sum += a[i];
double ave = sum / a.length;

reverse the values
within an array
for (int i = 0; i < N/2; i++)
{
    double temp = a[i];
    a[i] = a[N-1-i];
    a[N-1-i] = temp;
}

sort a sequence of
values in another array
double[] b = new double[N];
for (int i = 0; i < N; i++)
    b[i] = a[i];
StdArrays.sort(b);
```

Typical array processing code (for an array of type double values)

5.3 String Manipulation Methods

```
public class String (from string class)
{
    boolean[] toCharArray();

    int length();
    char charAt(int i);
    boolean[] getBytes(int i, int j);
    boolean contains(String substring);
    boolean startsWith(String pre);
    boolean endsWith(String post);
    int indexOf(String pattern);
    int indexOf(String pattern, int i);
    int compareTo(String t);
    String[] split(String delim);
    String toLowerCase();
    String toUpperCase();
    String trim();
    String replaceFirst(String s, String t);
    String[] getFields(String delimiter);
    boolean equals(Object t);
    int hashCode();
}
```

For the entire class see the documentation for string class

```
create a string with the same value as a
string or string-like representation, for example
a character array, a StringBuilder, a char
buffer, a StringBuilder, or a StringBuilder
representation of a character array
String s = new String();
String s = new String("Hello");
StdOut.print("Enter a string: ");
String s = StdIn.readLine();
StdOut.printf("%s %s %s", s, s, s);
StdOut.printf("%s %s %s", s, s, s);
StdOut.printf("%s %s %s", s, s, s);
```