CS 142 Midterm Cheat Sheet

```
for (initialization; test; update) {
    statement(s);
    ...
}
```

```
while (condition) {
    statement(s);
}
```

```
if (test) {
    statement(s);
} else if (test) {
    statement(s);
} else {
    statement(s);
}
```

```
public static type name(parameters) {
    statement(s);
    ...
    return expression;
}
```

```
type name = value;                          // variable declaration and initialization
Type objectName = new Type(parameters);     // object construction
```

Math Method	Description
`Math.abs(`*value*`)`	absolute value
`Math.min(`*v1, v2*`)`	smaller of two values
`Math.max(`*v1, v2*`)`	larger of two values
`Math.round(`*value*`)`	nearest whole number
`Math.sqrt(`*value*`)`	square root
`Math.pow(`*b, e*`)`	base to the exponent power

Random Method	Description
`nextInt(`*max*`)`	random integer from 0 to *max*-1

String Method	Description
`contains(`***str***`)`	true if this string contains the other's characters inside it
`endsWith(`***str***`)`, `startsWith(`***str***`)`	true if this string starts/ends with the other's characters
`equals(`***str***`)`	true if this string is the same as *str*
`equalsIgnoreCase(`***str***`)`	true if this string is the same as *str*, ignoring capitalization
`indexOf(`***str***`)`	index in this string where given string begins (-1 if not found)
`length()`	number of characters in this string
`replace(`***str1***`, `***str2***`)`	replace all occurrences in this string of *str1* with *str2*
`substring(`***i***`, `***j***`)`	characters in this string from index *i* (inclusive) to *j* (exclusive)
`toLowerCase()`, `toUpperCase()`	a new string with all lowercase or uppercase letters
`charAt(`***i***`)`	returns char at index *i*

Scanner Method	Description
`nextInt()`	reads/returns input token as `int`
`next()`	reads/returns input token as `String`
`nextDouble()`	reads/returns input token as `double`
`nextLine()`	reads/returns line as `String`
`hasNextInt()`	returns `true` if there is a next token and it can be read as an `int`
`hasNext()`	returns `true` if there is a next token to read
`hasNextDouble()`	returns `true` if there is a next token and it can be read as a `double`
`hasNextLine()`	returns `true` if there is a next line to read

Declaring and using Arrays

```
type[] name = new type[length];
name[index] = value;
```

ArrayList

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

`add(`**value**`)`	adds value to collection (appends at end of list)
`add(`**index, value**`)`	inserts given value at given index, shifting subsequent values right
`clear()`	removes all elements of the collection
`contains(`**value**`)`	returns `true` if the given value is found somewhere in this collection
`equals(`**collection**`)`	returns `true` if the given other collection contains the same elements
`get(`**index**`)`	returns the value at given index
`indexOf(`**value**`)`	returns first index where given value is found in list (-1 if not found)
`isEmpty()`	returns `true` if the collection has no elements
`remove(`**value**`)`	finds and removes the given value from this collection
`set(`**index, value**`)`	replaces value at given index with given value
`size()`	returns the number of elements in the collection
`toString()`	returns a string representation such as "[10, -2, 43]"

Classes

Field *(data inside each object)*

```
private type name;
```

Method *(behavior inside each object)*

```
public type name(parameters) {
    statements;
}
```

Constructor *(code to initialize new objects)*

```
public className(parameters) {
    statements;
}
```

toString method *(called when an object is printed)*

```
public String toString() {
    code that produces/returns a String;
}
```

Inheritance

```
public class name extends superclass {
}
```

Critter classes

```
public class name extends Critter {
    fields

    constructor

    public boolean eat() {
        statement(s) that return true (eat) or false (don't eat);
    }

    public Attack fight(String opponent) {
        returns either Attack.ROAR, Attack.POUNCE, or Attack.SCRATCH;
    }

    public Color getColor() {
        statement(s) that return a Color;
    }

    public Direction getMove() {
        statement(s) that return either Direction.NORTH, Direction.SOUTH,
        Direction.EAST, Direction.WEST, or Direction.CENTER;
    }

    public String toString() {
        statement(s) that return a String;
    }
}
```