CSE 142 Final Cheat Sheet

Syntax templates:

Declaring and using arrays

```
type[] name = new type[length];
type[] name = {VAL1, VAL2, VAL3, ...};
name[index] = value;
name.length   // number of elements in array
```

Declaring objects

```
Type objectName = new Type(parameters);
```

Classes

Field *(data inside each object)*

```
private type name;
```

Constructor *(code to initialize new objects)*

```
public className(parameters) {
   statement(s);
}
```

Method *(behavior inside each object)*

```
public type name(parameters) {
    statement(s);
}
```

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

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

Inheritance

```
public class name extends superclass {
   field(s), constructor(s), method(s), etc.
}
```

Critter class template:

```
public class name extends Critter {

    field(s)

    constructor(s)

    public Color getColor() {
        statement(s) that return a Color (e.g. Color.RED, Color.BLUE, Color.WHITE)
    }

    public Action getMove(CritterInfo info) {
        statement(s) that return Action.INFECT, Action.HOP, Action.LEFT, or Action.RIGHT
    }

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

CritterInfo Method	Description
`getFront(), getBack(), getLeft(), getRight()`	returns one of `Neighbor.WALL`, `Neighbor.EMPTY`, `Neighbor.SAME`, `Neighbor.OTHER`
`getDirection()`	returns one of `Direction.NORTH`, `Direction.SOUTH`, `Direction.EAST`, `Direction.WEST`

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

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

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

Scanner Method	Description
`nextInt(), hasNextInt()`	read/return input token as `int`; test if reading will succeed
`next(), hasNext()`	read/return input token as `String`; test if reading will succeed
`nextDouble(), hasNextDouble()`	read/return input token as `double`; test if reading will succeed
`nextLine(), hasNextLine()`	read/return line as `String`; test if reading will succeed

ArrayList Method	Description
`add(`***value***`)`	appends given value at end of list
`add(`***index***`, `***value***`)`	inserts given value at given index, shifting subsequent elements right
`get(`***index***`)`	returns value at given index
`remove(`***index***`)`	removes/returns value at given index, shifting subsequent elements left
`set(`***index***`, `***value***`)`	replaces value at given index with given value
`size()`	returns number of elements in list