Cram Review → AP Computer Science A

Quick Study Reference

- De Morgan's Law
  - Distributing the ! operator to negate a statement
  - Examples
    - `!(A || B)` → `!A && !B`
    - `!(A && B)` → `!A || !B`
- Generating a random number with the Math class
  - To generate a random int…
    - `int x = (int) (Math.random()*(`max-min`)+1)+`max`;`
  - To generate a random double…
    - `double y = (Math.random()*(`max-min`)+1)+`max`;`
- compareTo
  - `a.compareTo(b)` where `a` and `b` are strings
  - Subtracts the lexicographic code of `a` and `b`, `a-b`, then returns that value
- Types of errors
  - `NullPointerException`
    - When a method calls something that has a `null` value
  - Compile time error
    - Code does not run, crashes before it can go through the code
    - Syntax errors
    - Doesn't make it to runtime
  - Runtime error
    - Goes through the program but crashes in progress
    - Prints anything before it errors
- Searches
  - Sequential/Linear search
    - Goes through each element until desired element is found
    - Can be sorted *OR* unsorted
  - Binary search (Segment search)
    - Examines the middle element then checks the left segment if it is less or right if it is greater
    - MUST be a *sorted* array
  - Binary searches are faster than linear searches
- Sorting
  - Selection sort
    - Selecting a value and putting it into its appropriate position in the list
    - Could swap an index value with the greatest or smallest in the array or list
  - Insertion sort
    - Selects a value and compares it to the rest of the elements
    - Compares elements to the left of the selected element
- Boolean order of operations

    1) `!`
    2) `&&`
    3) `||`

- Constructors
    - If there are no constructors it will *not* error
        - If a class has no constructor in Java, the compiler will add a no-argument constructor
        - Creates a default empty constructor
    - Empty constructor
        - WILL NOT ERROR
        - A way to create an object without passing through specific parameters
        - Allows for the call `super();`
        - ALWAYS add an empty constructor when writing classes to avoid issues that may occur
    - Compile Time error
        - Occurs when a subclass has a no-argument constructor and variable is declared as `Class object = new SubClass();`
- Escape sequences
    - Allows certain actions in Strings
    - `\t` → Inserts a tab at the point of use
    - `\b` → Inserts a backspace at point of use
    - `\n` → Inserts a new line at point of use
    - `\r` → Inserts a carriage return in the text at the point of use
    - `\f` → Inserts a a form feed in the text at the point of use
    - `\'` → Inserts a single quotation (‘) at point of use
    - `\"` → Inserts a double quotation (“) at point of use
    - `\\` → Inserts a backslash (\) at point of use

- Dividing with `int`s and `double`s
    - `int / int`
        - Truncates and cuts off decimal (`int`)
    - `double / int`
        - `double`
    - `int / double`
        - `double`
    - `((double) int / int)`
        - `double`
    - `(double) (int / int)`
        - `double` but truncates because it is `int` division first
- Behavior vs. attributes
    - Attribute
        - Variables and instance variables
    - Behavior
        - Methods in a class
- Comparing objects
    - Object E == Object F
        - Checks if the objects point to the same place in memory
        - DOES NOT check if they have equal contents
- Overriding vs. overloading
    - Overriding
        - In a subclass and superclass
        - Same name, same parameter
        - Uses `@Overide` (not necessary, but practices good habits)
    - Overload
        - Same class
        - Same name, different parameters

- Common array algorithms
    - Max and min value
        - Minimum

```
int minIndex = 0;
for (int i = i; i < array.length; i++)
{
      if (array[i] < array[minIndex])
      {
            minIndex = i;
      }
}
```

    - Sum, average, or mode
        - For mode: use counter
        - Average

```
for (int i = 0; i < array.length; i++)
{
      //calculation here
}
return (double) sum / array.length;
```

    - Determining properties of a particular property
        - Properties of a value

```
int counter = 0;
for (int i = 0; i < array.length -1; i++)
{
      if (array[i].equals("property goes here")
      {
            counter++;
      }
}
```

    - Access consecutive pairs of elements
        - Check first number and if it is equal to the second, it's a consecutive pair
        - Consecutive

```
boolean consecutive = false;
for (int i = 0; i < array.length -1; i++)
{
      if (array[i] == array[i+1])
      {
            consecutive = true;
      }
}
```