CMSC131

Overloading, Ternary Operator, Switch Statement
You can "overload" a method name by having multiple implementations of a method with different parameter lists.

Consider these:

```java
public int fun1(int i, int j);
public int fun1(int i, float f);
public int fun1(float f, int i);
public int fun1(Double d, Integer i);
```
Method Overloading Limitations

You cannot have overloaded methods differ by only the return type or only by different local names within parameter lists.

These conflict:

```java
public int fun1(int i, int j);
public boolean fun1(int i, int j);
```

These conflict:

```java
public int fun1(int i, int j);
public int fun1(int a, int b);
```
The ternary operator

The ternary operator is of the form

\[(\text{boolean_expression})?\text{if}\_\text{true}:\text{if}\_\text{false};\]

A simple example using assignment

\[\text{String } s=(x<0)\text{?}"\text{Negative}":"\text{Not Negative}";\]

More useful "tricks" include things such as

\[
\text{minVal} = (a < b) \ ? \ a : b;
\]

\[
\text{absValue} = (a < 0) \ ? \ -a : a;
\]
The switch statement

Basically, a switch means that for simple primitives, instead of:

```java
if (option == 1) {code1;}
else if (option == 2) {code2;}
else if (option == 3) {code3;}
```

you can use:

```java
switch (option) {
    case 1: code1; break;
    case 2: code2; break;
    case 3: code3; break;
}
```