Introduction to java

COMP1030
Lecture #4

Housekeeping

- Our goal - 100% pass rate in this course.
- Review the lecture slides ahead of time.
- Review the lecture slides after class with a study group.
- Repeat the lab at home 1-2 times.
- Take notes during class.
- Weekly optional tutorials take place every week on Wednesdays, from 3-4pm in room K323.

Review

- Instance variables
- setters/getters
- Method Signatures
- Constructors
- Main method
- String class

Pseudocode

- Pseudocode is an informal, english-like language.
- Assists in the development of coding solutions without concern for syntax.
- Helps you "think out" a programming solution to a given problem.

Pseudocode

pseudocode:

Display the result of the calculation in the terminal window

Java code

```
System.out.println("Total:" totalPrice);
```

Flow Charts

- A graphical representation of a process or flow of a program.

Flow Chart Symbol	Meaning	Explanation
	Start and end	The symbol denoting the beginning and end of the flow chart.
	Step	This symbol shows that the user performs a task. (Note: In many flow charts steps and actions are interchangeable.)
	Decision	This symbol represents a point where a decision is made.
	Action	This symbol means that the user performs an action. (Note: In many flow charts steps and actions are interchangeable.)
	Flow line	A line that connects the various symbols in an ordered way.

relational operators

Operator	Name	Example expression	Meaning
`==`	Equal to	`x == y`	`true` if `x` equals `y`, otherwise `false`
`!=`	Not equal to	`x != y`	`true` if `x` is not equal to `y`, otherwise `false`
`>`	Greater than	`x > y`	`true` if `x` is greater than `y`, otherwise `false`
`<`	Less than	`x < y`	`true` if `x` is less than `y`, otherwise `false`
`>=`	Greater than or equal to	`x >= y`	`true` if `x` is greater than or equal to `y`, otherwise `false`
`<=`	Less than or equal to	`x <= y`	`true` if `x` is less than or equal to `y`, otherwise `false`

Java Boolean operators

Operator	Operation
++ --	increment, decrement
+ -	unary plus, minus
!	boolean not
(<type>)	cast to <type>
* / %	multiplication, division, remainder
+ -	addition/concatenation, subtraction
< <= > >=	relational ordering
== !=	relational equality, inequality
&&	boolean and
||	boolean or
= += -= *= /= %=	assignments

Control Structures

- Java control structures enable the programmer to specify what statement(s) are executed based upon some condition.

if statement (single selection)

- Selection statements are used in programming to choose among alternative courses of action.

if statement (single selection)

pseudocode

If a students grade is greater than or equal to 60

```
        print "passed"
```

java code

```
        if (studentGrade >=60)
                System.out.println ("passed");
```

If-else statement (double selection)

pseudocode

```
If a students grade is > than or = to 60
        print "passed"
otherwise
        print "failed"
```

java code

```
        if (studentGrade >=60)
                System.out.println ("passed");
        else
                System.out.println("failed");
```

nested if-else statements

pseudocode

```
If a students grade is greater than or equal to 90
        print "A"
otherwise
        If a students grade is greater than or equal to 80
                print "B"
        otherwise
                If a students grade is greater than or equal to 70
                        print "C"
```

nested if-else statements

java code

```
if (studentGrade >= 90)
        System.out.println("A");
else
        if (studentGrade >= 80)
                System.out.println("B");
        else
                if (studentGrade >= 70)
                        System.out.println("C");
```

Using blocks of code

- To include several statements in the body of an if statement one can use a block of code.

```
if (studentGrade >= 90)
        System.out.println("A");
else
        {
                System.out.println("you scored less than an A");
                System.out.println("Please re-do the exam");
        }
```

while repetition statement

- A repetition statement allows you to specify that a section of code should repeat while some condition remains true.

pseudocode

```
As long as the count is less than 10
        keep counting
```

java code

```
int count = 1:
while (count <11)
{
        System.out.println("count is:" + count);
        count= count +1;
}
```

increment and decrement operators

- ++ increment
- -- decrement

- ++a prefix increment (increment by 1, then use the new value of a in the expression in which it resides)

- a++ postfix increment (use the current value of a in the expression in which it resides, then increment a by 1)

increment and decrement operators

```
int a = 5;
        System.out.println("a is now", a++);
```

prints 5

```
int b = 5;
        System.out.println("b is now", ++b);
```

prints 6