CSCB07

Albion Fung ~(._.)~
albion@amacss.org

Software Version Control (SVN)

- Provides a way for different people to work together
- Organize different versions
- Track back to older versions
- See changes

- svn add
- svn checkout
- svn commit
- svn update
- svn commit
- svn delete
- svn status
- svn revert
- svn log -r 1:HEAD

svn checkout link

svn status

- ?: untracked
- A: Added but not committed yet
- M: modified and not committed yet
- C: conflict!

conflicts

- I don’t get it either.
- jk I figured it out

Conflicts

- On initial conflict, svn shows you the differences
- Either postpone it to solve later or do it now

- svn delete: removes a file
- svn log: shows you the commit logs
- svn revert: revert changes to most recent commit
- svn update -r# filename: “update” file name to revision #


Java™

Primitive types

- int (and long, short, etc)
- double (and float)
- boolean
- char

String is NOT a primitive type

- String a = “hi”; // immutable object
- String a = new String(“hi”); // object

Wrapping

- Basically an object version of primitives
- Why? I don’t fucking know
- Have some useful methods… that I never use
- Capitalize first letter and make it full word
- int -> Integer
- double -> Double

IMO totally useless but hey

Commenting

- // blah blah
- /*black sheep*/
- /**i forgot the next line*/ //JavaDocs

Java docs

- use /** */
- @param paramname paramdesc
- @author name
- @return desc

```
/**
 * A method that returns the number of pieces of chocolates I have yet to eat but I will.
 * @author Albion
 * @param name Name of the person
 * @param choc Type of chocolate
 * @return number of pieces
```

Java methods and inheritance

Casting

- Son extends Dad
- Dad d = new son(); // a okay
- Son s = new Dad(); // not okay, son is not a dad (thank god)
- Son so = (Son)d;
- Dad joke = new Mom(); // not okay, her jokes are shit
- Son son = new Son();
- Dad what = (Dad)son; // not okay, the son was not a dad to begin with

Abstract

- Tfw you just need a mold
- abstract prefix
- If inherited, must have a physical implementation in the child
- Abstract classes may have some implemented methods

Interfaces

- Like abstract class, except no implementation at all

- private, public and protected inheritance rules

Generics

- Data<type, type,…>
- Basically to guarantee correct return type
- ArrayList<String> a = new ArrayList<String>();

ANY. QUESTIONS.?