Tutorial 2: Simple Java Programming

COMP 202: Intro to Computing 1

Winter 2009

TA: Robert Rolnick
E-mail: Robert.Rolnick@mail.mcgill.ca

Tutorial 2: Simple Java Programming

1. Syntactic and Semantic Errors
2. Expressions and Data Types
3. Conversions
4. Writing Simple Programs
5. Notes on Assignments

Syntactic Errors

- Syntactic errors are caught by the compiler.
- Produce consistent compile-time errors.
- Depending on the language syntactic errors can be caused by case-sensitivity, white space, line delimiting, etc.
- The compiler does not know what you mean. Therefore, it might find the error on a different line than where it occurs.

"Programs must be written for people to read, and only incidentally for machines to execute."
~Structure and Interpretation of Computer Programs

Syntactic Errors: Example

- The following code contains 4 errors:

```
public class SimpleClass
{
  public static viod main(string[] args)
  {
    system.out.println("Hello World"):
  }
}
```

- What happens when we compile it?

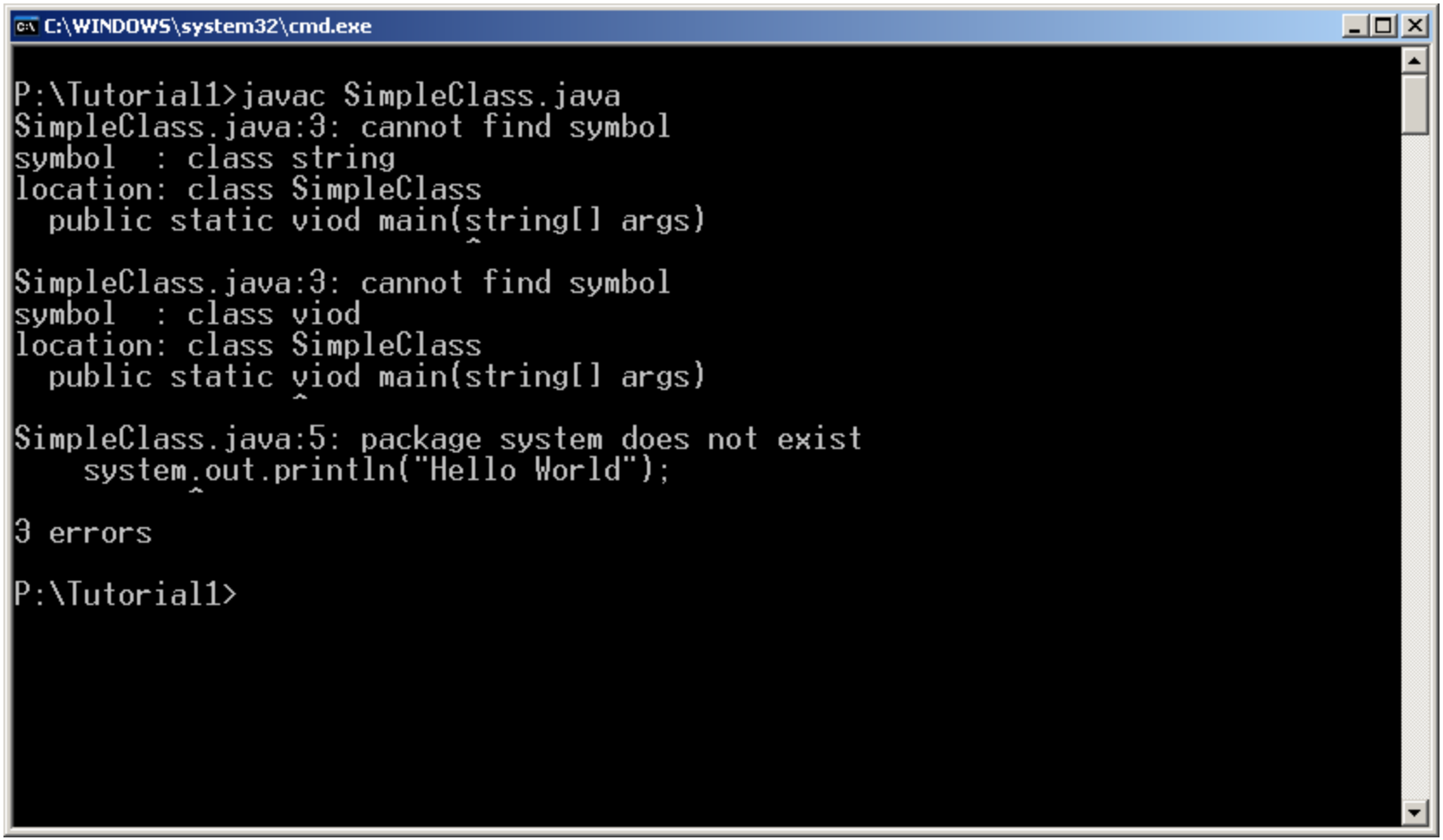
Syntactic Errors: Example (Continued)

Syntactic Errors: Example (Continued)

- The compiler points out where it *noticed* the error: **[line: 5]**
- The compiler points out what it thinks the error is: **';' expected**
- Let's fix this error and recompile.

```
public class SimpleClass
{
  public static viod main(string[] args)
  {
    system.out.println("Hello World");
  }
}
```

Syntactic Errors: Example (Continued)



Syntactic Errors: Example (Continued)

- The compiler found the remaining three errors, and classified them as either: "**cannot find symbol**" or "**cannot find package**"
- Cannot find symbol errors occur by incorrectly referencing a variable/keyword, or by referencing a variable that doesn't exist. I misspelled the keyword void. Due to case sensitivity, I misspelled String as well.
- Many packages are built into Java, they provide functionality like displaying text. By referencing a package incorrectly, you get a cannot find package error. (I typed system instead of System.)
- Fixing these three errors is left as an exercise for the reader.

Semantic Errors

- Semantic errors are NOT ALWAYS caught by the compiler.
- Sometimes the compiler will catch a semantic error. For example, when you try to print an uninitialized variable.
- Other times the program will compile fine, and give errors on execution. This may result in a crash (e.g. divide by 0), an infinite loop, or incorrect output.
- These types of errors are sometimes called logic errors, as they caused the program to run incorrectly.

"Computers are good at following instructions, but not at reading your mind."
~Donald Knuth

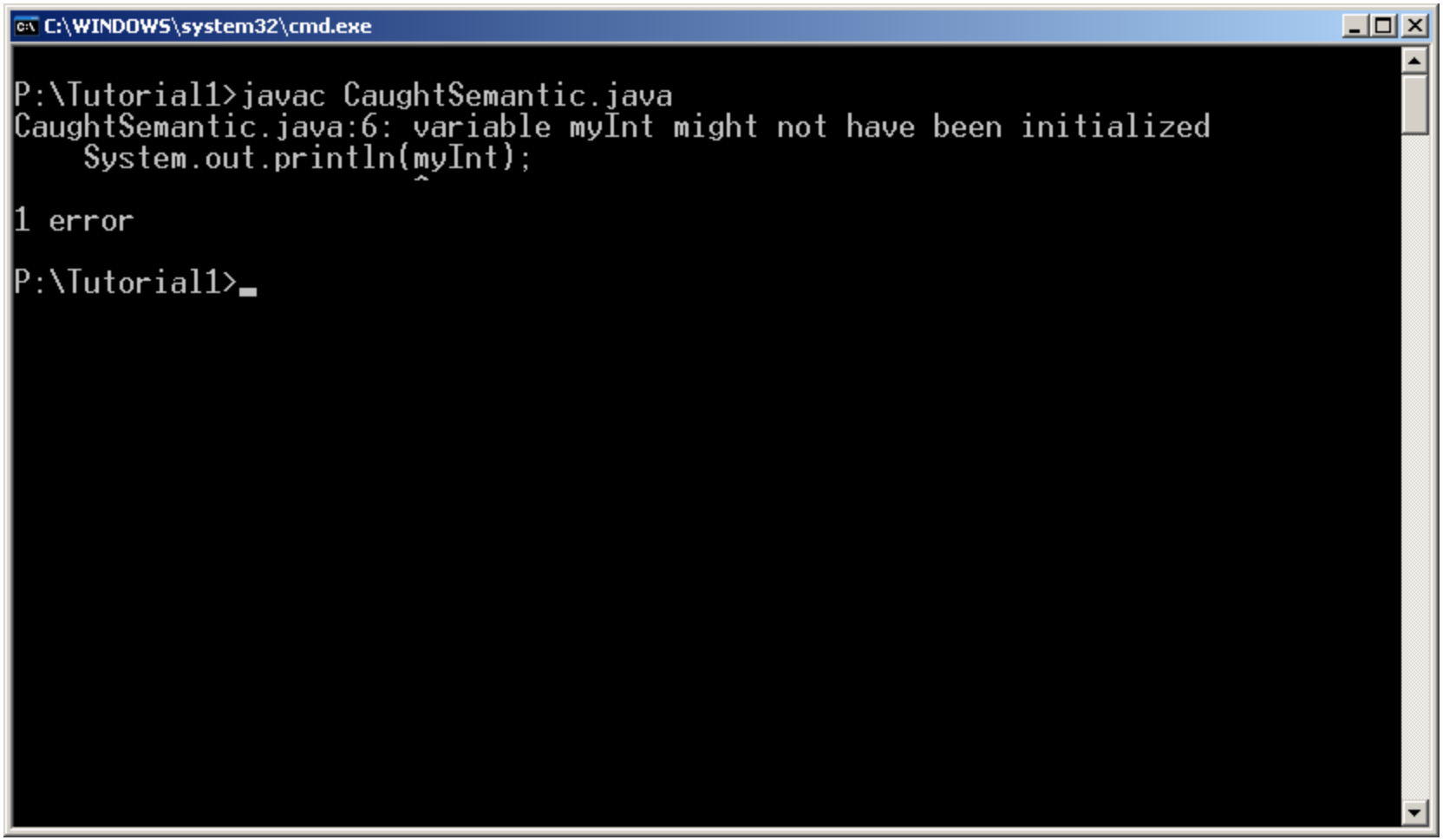
Semantic Errors: Example 1 (Continued)

- Some semantic errors are caught by the compiler.
- Consider the following code:

```
public class CaughtSemantic
{
  public static void main(String[] args)
  {
     int myInt;
     System.out.println(myInt);
  }
}
```

- What happens when we compile it?

Semantic Errors: Example 1 (Continued)



Semantic Errors: Example 2

- Some semantic errors only show up at runtime.

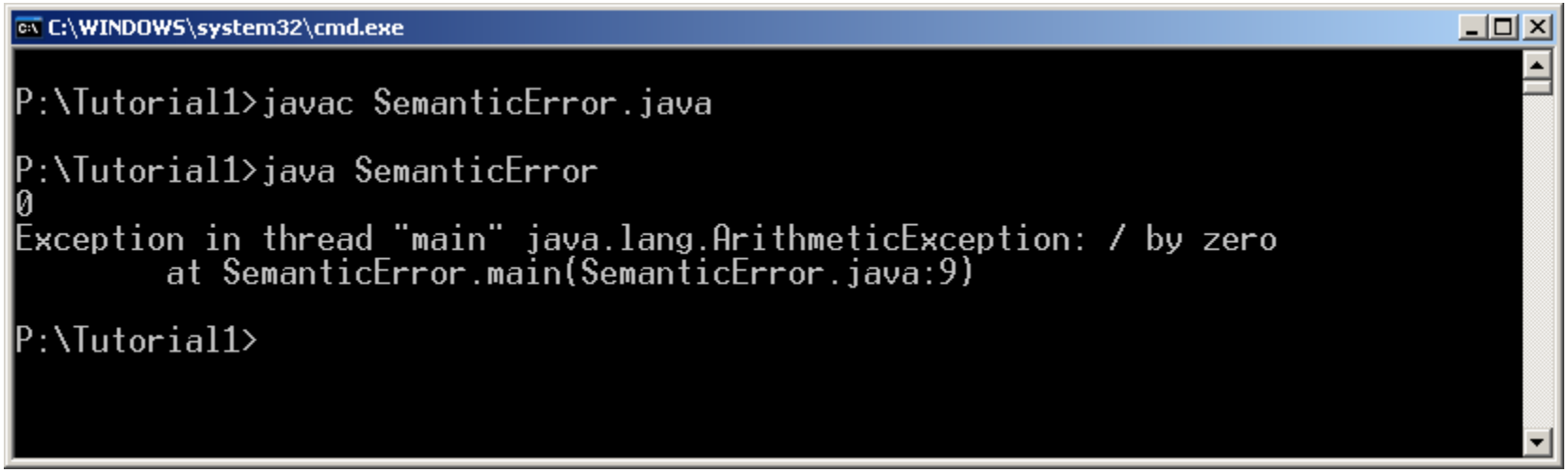
- Consider the following code:

```
import java.util.Scanner;

public class SemanticError
{
  public static void main(String[] args)
  {
    Scanner keyboard = new Scanner(System.in);
    int x = keyboard.nextInt();
    System.out.println("5 / " + x + " is: " + (5/x));
  }
}
```

Semantic Errors: Example 2 (Continued)

- The code is syntactically correct, and 0 is a valid integer. But when I divide 5 by 0, I crash.



- Not all semantic errors lead to crashes.

Semantic Errors: Common Causes

- Some of the main causes of semantic errors in programming are:
  - Division by 0.
  - Incorrect bounds on comparisons. (Off by 1 errors.)
  - Integer division. (What does 5/2 equal?)
  - Incorrect exit conditions on loops.
  - No exit condition in recursion (much later in the course.)
  - Not validating user-input. (Not a heavy focus of course.)

Debugging

- Debuggers let you do the following:
  - Step through your code one line at a time. (Starting at a specified point.)
  - See the state of your variables.
  - See the call stack. (Useful for function calls, and recursion.)
- Advanced debuggers, such as Eclipse's debugger, allow:
  - Only break on a line when a specified condition is true.
  - Change the state of a variable while debugging.
- For now, we will deal exclusively with "print-line" debugging.
- Using Eclipse's debugger will be covered in Tutorial 4.

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write your code as cleverly as possible, you are, by definition, not smart enough to debug it."

~Brian Kernighan

How Does Print-Line Debugging Work?

- Print-Line debugging works by placing *System.out.println()* calls throughout your code.

- These calls serve two purposes:
  1. Sanity checks to ensure a specific line of code is hit.
  2. Echoing back the current value of your variables.

- Sanity checks are checkpoints in your code. For example:

```
System.out.println("Reached point FOO in the code.");
```

- Echoing back a variable's current value is done as follows:

```
System.out.println("bar = " + bar);
```

Expressions

- Expressions in Java are constructs made up of three components.
    1. Variables / Constants
    2. Operators
    3. Method Invocation (later in the course)

- Expressions return values. These values can be a fundamental data type (int, float, double, etc.) or an object (later in the course.)

- In the same way that arithmetic follows an order of operations (PEMDAS), so too do complex expressions.

"Mathematicians and programmers differ. The former sees x=x+1 and says 'No it doesn't.'"
~Paraphrased from an episode of DotNetRocks

Expressions: Example 1

- In the code below, the expression is written in **bold red text**.

```
int x = 5;
```

- This expression contains a variable: x
- This expressions contains an (assignment) operator: =
- This expression contains a constant: 5
- This expression return a value of type int.

Expressions: Example 2

- Let's try a slightly harder example.

```
int x = 5;
int y = 2;
boolean z = (x == y);
```

- The third expression contains three variables: x, y, and z.
- The third expression contains an (assignment) operator: =
- The third expression contains an (equality) operator: ==
- The third expression return a value of type boolean.
- You can also consider the third line to be made up of two expressions. The first (x == y), returns a boolean. The second, z = false, returns a boolean as well.

Variables

- Variables can represent both fundamental data types and objects.
- Variable names are case-sensitive.
- Variable names can begin with a letter, _, or $. Subsequent characters can also be numbers.
- Reserved words and keywords cannot be used as variables.
- Variable naming convention exists. (e.g. Don't use a $ in a variable name, use camelCase notation, and make the names descriptive.)

"One man's constant is another man's variable."
~Alan Perlis

Variables

- The following list of words cannot be used as variable names:

• abstract	• else	• interface	• switch
• assert	• enum	• long	• synchronized
• boolean	• extends	• native	• this
• break	• false	• new	• throw
• byte	• final	• null	• throws
• case	• finally	• package	• transient
• catch	• float	• private	• true
• char	• for	• protected	• try
• class	• goto	• public	• void
• const	• if	• return	• volatile
• continue	• implements	• short	• while
• default	• import	• static	
• do	• instanceof	• strictfp	
• double	• int	• super	

http://java.sun.com/docs/books/tutorial/java/nutsandbolts/_keywords.html

Data Types & Operators

- Java contains eight (8) fundamental data types:
  1. boolean
  2. byte
  3. char
  4. double
  5. float
  6. int
  7. long
  8. short
- Java contains eleven (11) types of operators:
  1. additive
  2. assignment
  3. bitwise
  4. equality
  5. logical
  6. multiplicative
  7. postfix
  8. relational
  9. shift
  10. ternary
  11. unary

"Premature optimization is the root of all evil in programming."
~Made famous by Donald Knuth, but originally from Charles Antony Richard Hoare

Data Types

- All of the data types use lower case naming.
- Java is case-sensitive, so Double and double are not the same. While your code may generally work, it is incorrect and may fail in unexpected scenarios.
- booleans can only store true or false.
- chars can store unsigned integral values (or a single character).
- byte, short, int, and long can store signed integral values.
- float and double can store numbers with decimal places.

Data Types: Sizes

- The following chart shows the various data types in Java.
- Compilers give default values to variables when used as fields (class variables), but this should not be relied upon.

	byte	short	int	long	float	double	boolean	char
# Bits:	8	16	32	64	32	64	8	16
Min:	-2^7	-2^{15}	-2^{31}	-2^{63}	Single-Precision IEEE 754 (another class)	Double-Precision IEEE 754 (another class)	N/A	\u0000
Max:	$2^7 - 1$	$2^{15} - 1$	$2^{31} - 1$	$2^{63} - 1$				\uffff
Default:	0	0	0	0L	0.0f	0.0d	false	\u0000

- Although booleans can only be true or false, since memory is byte addressable, they often need 8 bits instead of only 1.
- Chars are unsigned, they represent a 16-bit unicode character.

Operators: Precedence

- Operators have a set precedence. It is:

Operator	Examples
Postfix	x++ x--
Unary	++x ++x +x -x ~ !
Multiplicative	* / %
Additive	+ –
Shift	<< >> >>>
Relational	< > <= >= instanceof
Equality	== !=
Bitwise AND	&
Bitwise XOR	^
Bitwise OR	|
Logical AND	&&
Logical OR	||
Ternary	?:
Assignment	= += –= *= /= %= &= ^= |= <<= >>= >>>=

Statements

- Statements form executable units. They come in five (5) types:

1. Assignment
2. Increment / Decrement
3. Control Flow (later in the course)
4. Method invocation (later in the course)
5. Object Creation (later in the course)

"Weeks of programming can save hours of planning."
~Anonymous

Statements

- Comments come in two forms.
  1. Single line comments: **//this is a comment**
  2. Multiple line comments: **/* this is a comment too, even if it spans multiple lines. */**

- Comments help explain code, both to others and to yourself.

- Good comments explain the "why" of code not the "how."

- Missing comments are bad. Incorrect (or outdated) comments are worse.

"When code and comments disagree, both are probably wrong."
~Norm Schryer

What is a Conversion?

- Conversions transform variables from one data type into another data type. For example, they can transform a short into an int.

- Conversions can occur in three ways:
    1. Assignment
    2. Promotion
    3. Casting

- There are two types of conversions:
    1. Widening Conversions
    2. Narrowing Conversions

"There are two kinds of languages: the ones people complain about, and the ones nobody uses."
~Bjarne Stroustrup

Assignment Conversions

- An assignment conversion occurs when a value of one type is assigned to a variable of another type. For example:

```
public class AssignmentConversion
{
  public static void main(String[] args)
  {
    int x = 5;
    float y = x;
    System.out.println(y);
  }
}
```

- In the above code, x is an int. However, it is being assigned to a float. As such, y will contain the value 5.0.
- If you assigned a float to an int the compiler gives an error!

Promotion Conversions

- Promotion conversions occur due to operators and operands.

```
public class Average
{
  public static void main(String[] args)
  {
    float sum = 95.0 + 80.5 + 87.5;
    int count = 3;
    float avg = sum / count;
    System.out.println(avg);
  }
}
```

- Count is an int, but to do the division it gets promoted to a float.
- The promotion happens due to the arithmetic not the assignment! That is to say “float avg = 5/2”; will NOT put 2.5 in avg!

Casting Conversions

- Casting is an explicit conversion between types. If Java can convert between two types, it'll let you do it by a cast.

```
public class TypeCastExample
{
  public static void main(String[] args)
  {
    double fSum = 2.5;
    int iSum = (int)fSum;
    System.out.println(iSum);
  }
}
```

- By typecasting double to int we truncate the decimal.
- WATCH OUT: Once you learn about objects, you might begin writing casts that will compile, and then crash at runtime.

Widening Conversions

- Widening conversions are *usually* lossless. However, they might round! (Try casting a large long value to a float.)
- The following are the types of widening conversions in Java:

From	To
byte	short, int, long, float, or double
short	int, long, float, or double
char	int, long, float, or double
int	long, float, or double
long	float or double
float	double

"Beware of bugs in the above code; I have only proved it correct, not tried it."
~Donald Knuth

Narrowing Conversions

- Narrowing conversions run the risk of losing data.
- Watch out when converting a byte or short to a char! The char is unsigned, so the value you read might not be the value you put in!
- The following narrowing conversions in Java require a cast:

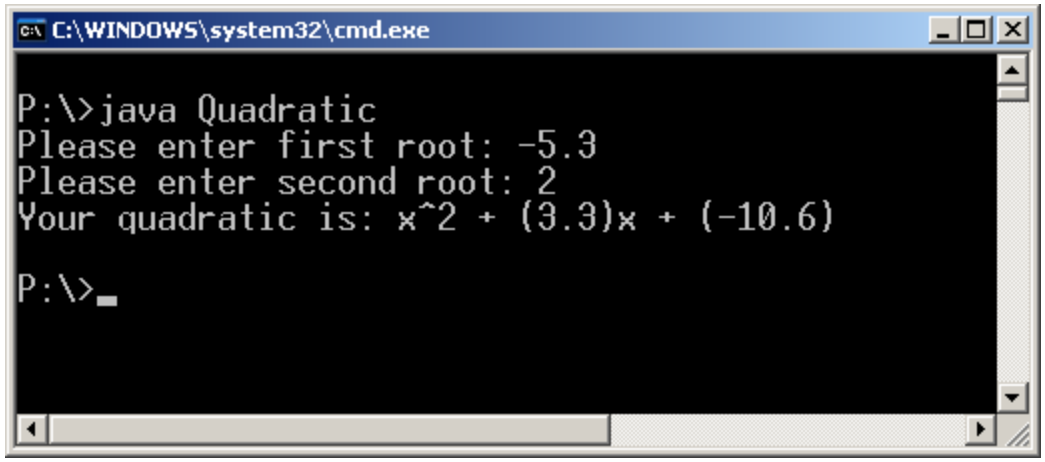
From	To
byte	char
short	byte or char
char	byte or short
int	byte, short, or char
long	byte, short, char, or int
float	Byte, short, char, int, or long
double	byte, short, char, int, long, or float

"Simplicity is prerequisite for reliability."
~Edsger W. Dijkstra

Quadratics

- Write a program with takes in the roots of a quadratic, and outputs the general form 2nd degree polynomial with those roots.

- For example: If you were to supply the inputs -5.3 and 2 the output will be: "Your quadratic is: x^2 + (3.3)x + (-10.6)"



"Owning a computer without programming is like having a kitchen and using only the microwave oven."
~Charles Petzold

Heron's Formula

- Heron's Formula gets the area (A) of a $\triangle$ from its legs (a, b, c).
- The formula is: $s = (a + b + c)/2$ $A = \sqrt{s(s-a)(s-b)(s-c)}$
- Write Heron's formula in Java.
- The code to take a square root is shown below.

```
import java.lang.Math;
public class SquareRoot
{
  public static void main(String[] args)
  {
    double root = Math.sqrt(9);
  }
}
```

"Measuring programming progress by lines of code is like measuring aircraft building progress by weight."
~Bill Gates

General Assignment Advice

- Start on the assignments early. Later ones do get harder.
- Ask clarifying questions on MyCourses so everyone can benefit.
- Alternatively, TAs can help 1-on-1 during their office hours.
- The warm-up questions cover the same material as the graded problems. As such, they serve as excellent building blocks.
- In order to grade your code, TAs need to read it. Clear variable names, proper structure, and comments lead to happier TAs.

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live."

~ Rick Osborne

Specific Tips for Assignment #1

- Hopefully you've already started Assignment #1.
- This assignment deals with course policies, and Java basics.
- Don't worry if you don't understand why you wrote all your code.
- Its difficulty should come solely from learning the commands and syntax. Once the commands are learnt, the order to use them should be fairly obvious.
- This semester (Winter 2009), the questions 3 and 4 each have two possible solutions. Both will be accepted for full credit.

"It always takes longer than you expect, even when you take into account Hofstadter's Law."
~ Douglas R. Hofstadter

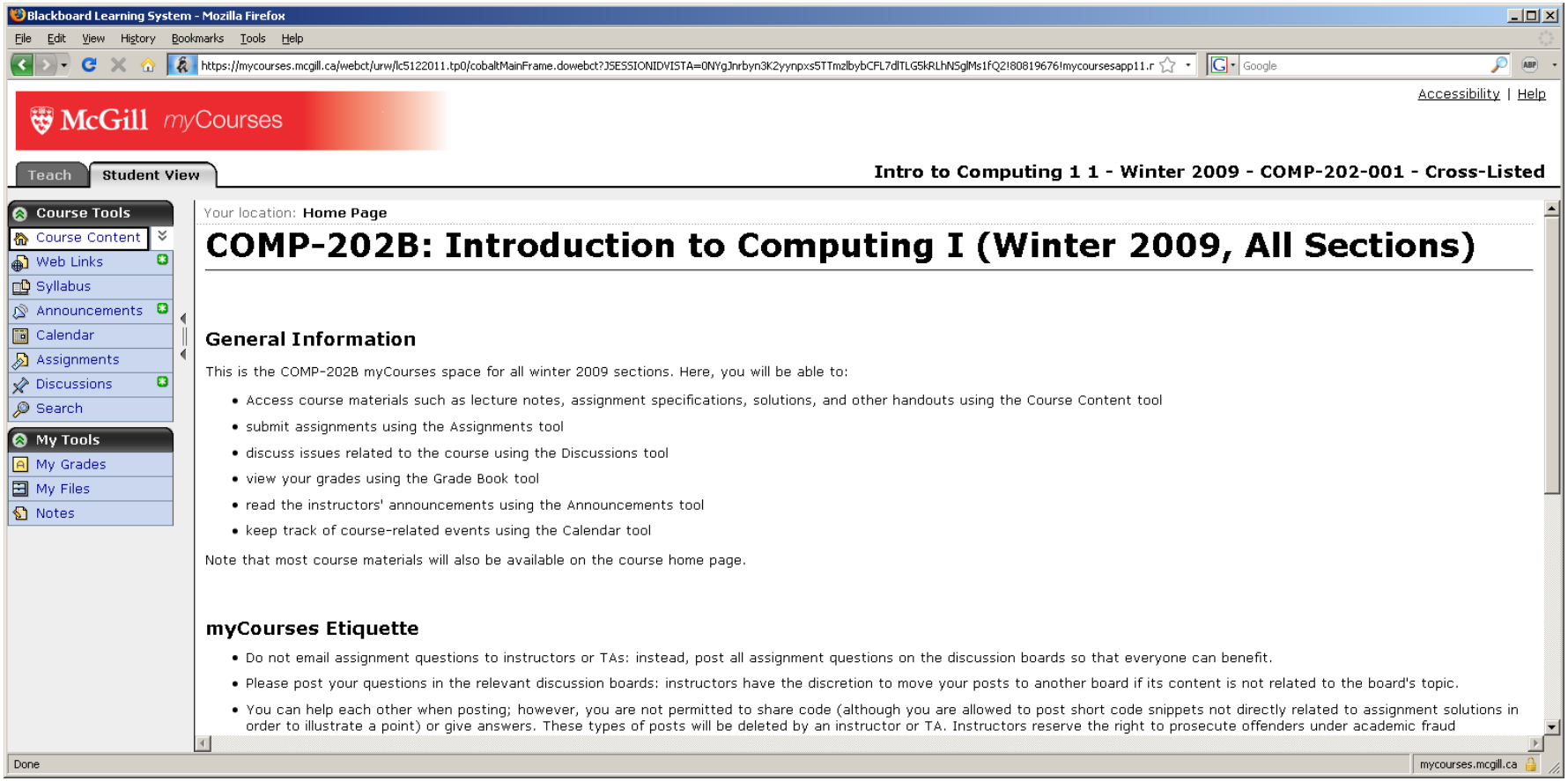
Submitting Assignments

- Assignments can be submitted as follows:
  1. Log onto the Comp-202 section of myCourses.
  2. Click on “Assignments” on the left menu bar.
  3. Click on the assignment drop box you want to use.
  4. Click the “Add Attachments” button.
  5. Click on “My Computer.”
  6. Click on the “Browse” buttons.
  7. Upload files.
  8. Click “OK.”
  9. Click “Submit” then “OK” on the confirmation dialog.
- Comments and submission notes are optional.

“The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time.” *~Tom Cargill*

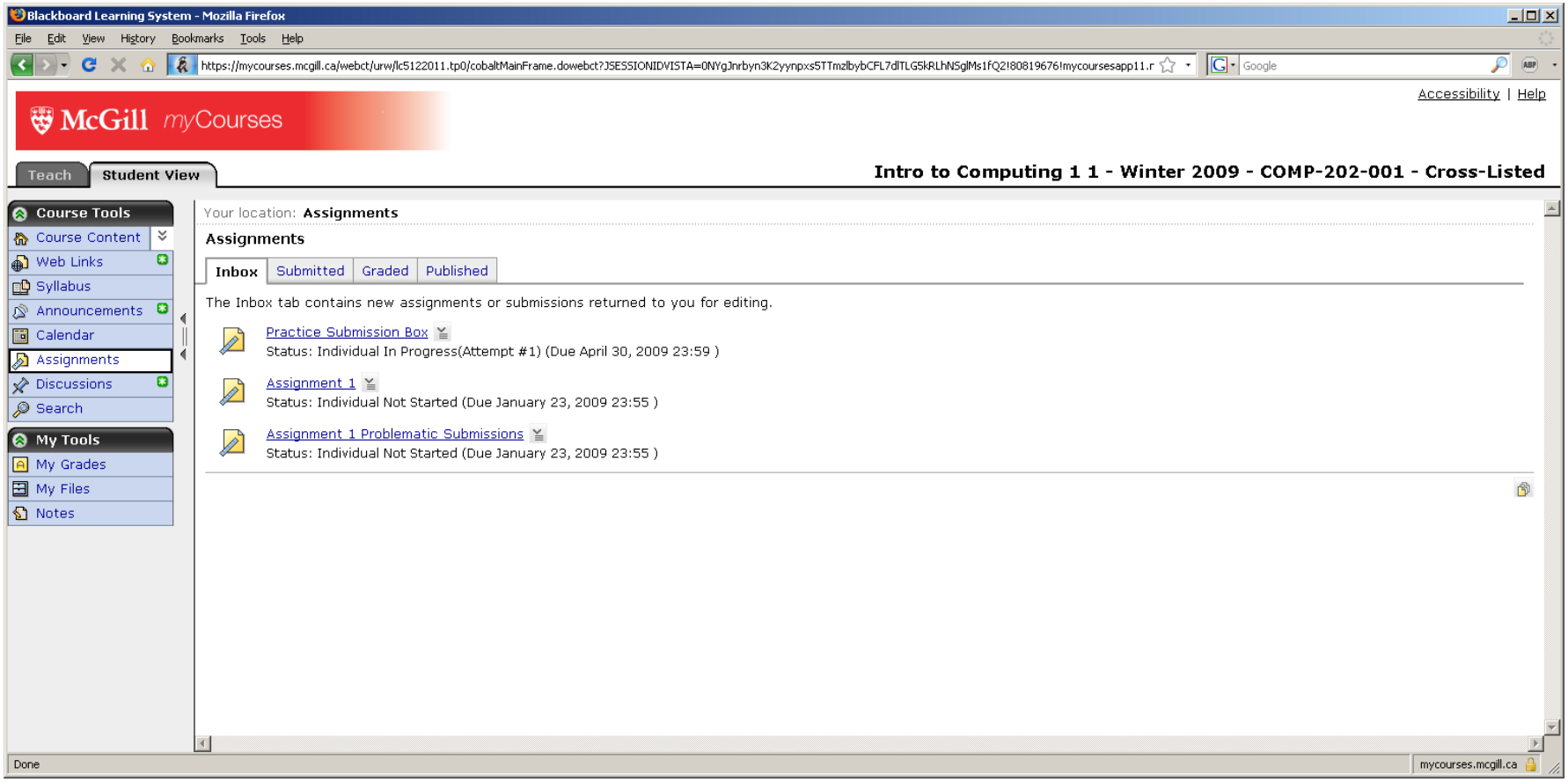
Submitting Assignments (Continued)

1. Log onto the Comp-202 section of myCourses.



Submitting Assignments (Continued)

2. Click on “Assignments” on the left menu bar.

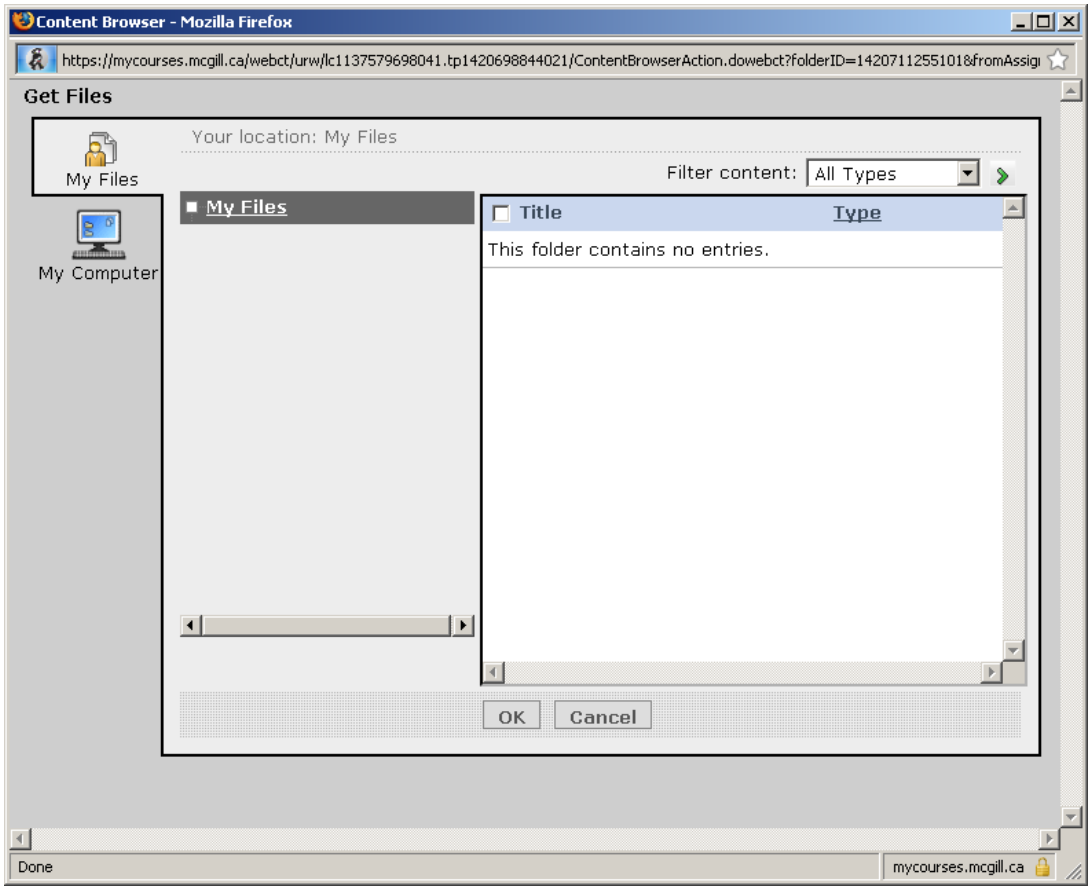


Submitting Assignments (Continued)

3. Click on the assignment drop box you want to use.

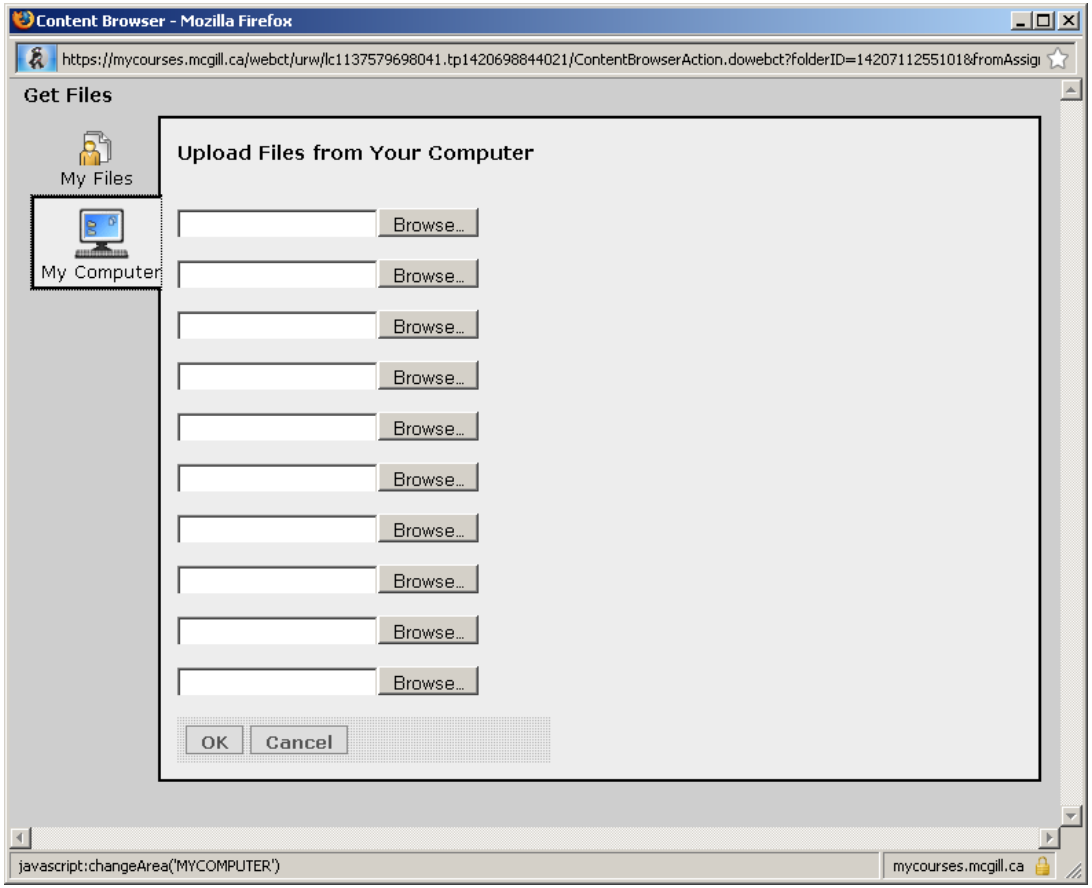
Submitting Assignments (Continued)

4. Click the “Add Attachments” button.



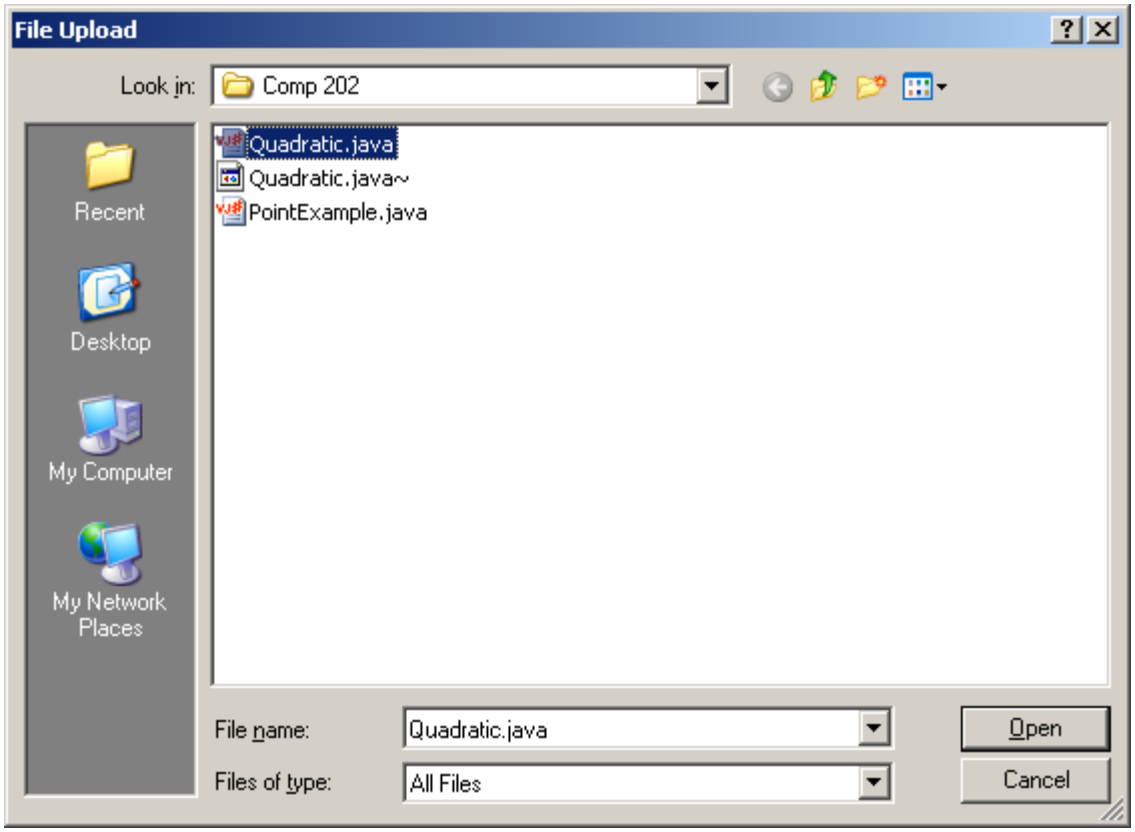
Submitting Assignments (Continued)

5. Click on “My Computer.”



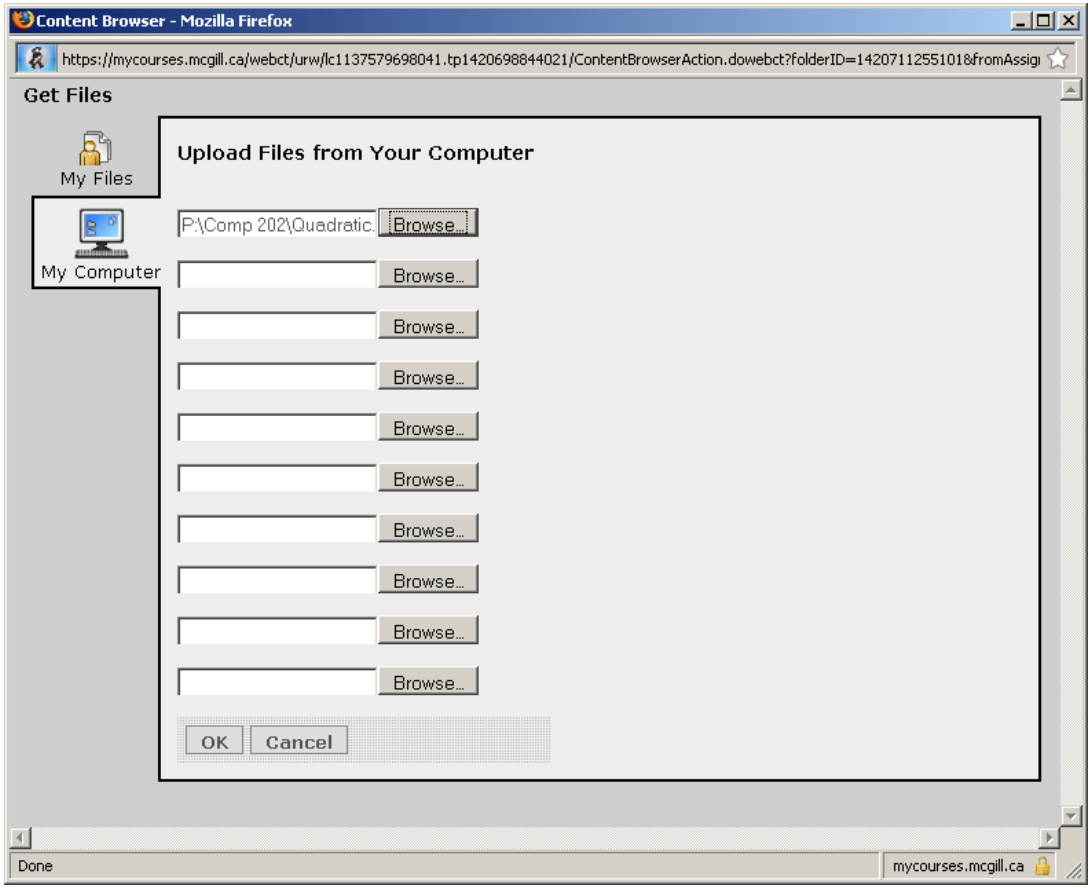
Submitting Assignments (Continued)

6. Click on the "Browse" buttons. (Note: .java~ files are back-up files created by some programs – they should not be uploaded.)



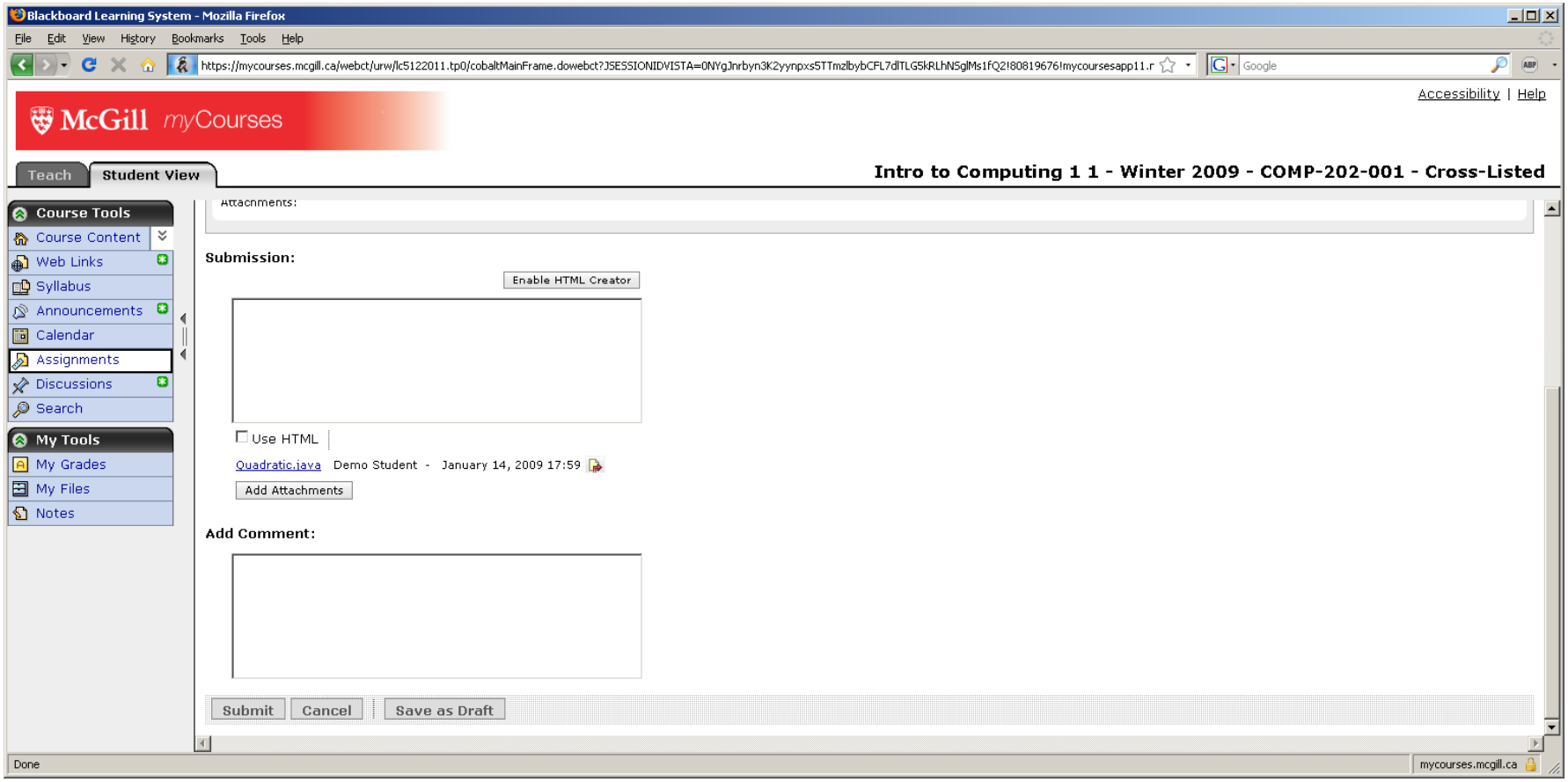
Submitting Assignments (Continued)

7. Upload files.



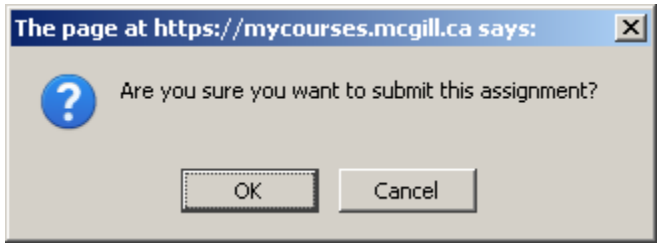
Submitting Assignments (Continued)

8. Click “OK.”

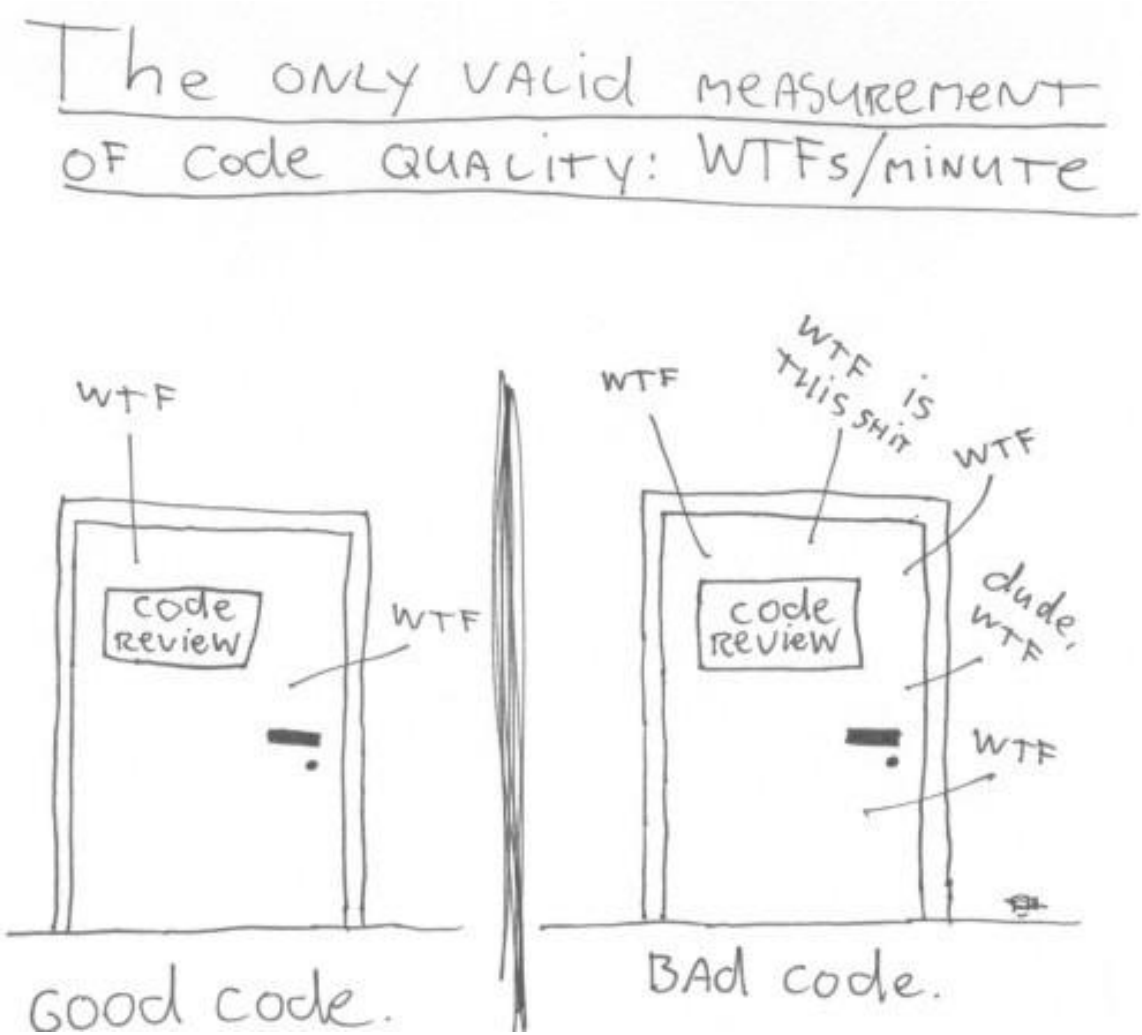


Submitting Assignments (Continued)

9. Click “Submit” then “OK” on the confirmation dialog.



Questions?


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF is THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.