GEOG 178/258:
Conceptual Modeling and Programming for the Geo-Sciences

TA: Rui Zhu
Week 2: Debug in Eclipse and Java Basics
Part I: Debug in Eclipse
1. Warnings or errors in your code

Common errors made by beginners

- No semicolon after statements;
- Brackets, parenthesis, quotes are singly used;
- Messed up with names of variables, classes and methods (e.g. capitalization);
...
2. Check the logic of your code

- Change to the Debug perspective in Eclipse:
• Create break point(s):
  A breakpoint is a location in your source code that will halt execution so that you can view the information about a program's state.

➢ Right-click along the left side of the java editor on the line of code → select Toggle Breakpoint;
➢ Double-click on the left side of editor

```
10 Scanner scanner = new Scanner (System.in);
11 int n=6;
12 do {
13     System.out.print("Enter a number (n>2): ");
14     n=scanner.nextInt();
15 }while (n<2);
```
• Watch expressions

Use it to keep track of complex expressions.
• Inspect variables

Use it to track the value of variables
Part II: Java Basics
Basic elements in java code

- **Class** - A class can be defined as a template/ blue print that describes the behaviours/states that object of its type support. *Example: the Polyline{} is a class*

- **Object** - Objects have states and behaviours. Example: *A polyline has states - startpoint, endpoint, name as well as behaviours – calculate its length, print its name*. An object is an instance of a class.

- **Methods** - A method is basically a behaviour. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed. *Example: a polyline can calculate its length; length() is the method of Polyline{}.*

- **Instance Variables** - Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables. *Example: a Polyline's name is “Hollister”.*
Call-by-reference and Call-by-value

- Types of variables
  - primitive data type (int, double, char, float...)
    - value is passed to the new variable of the called method
  - reference data type (Double, Point, Polygon...)
    - object is passed to the new variable of the called method
**Primitive data type and Reference data type**

Primitive data type holds the actual data.

Reference data type holds the address of the object it referred to.

```
Number a;

Memory
```

```
Number a;
    a = new Number(34);

Memory
```

The actual object

---

Rui Zhu  
GEOG 178/258  
Winter 2016
Call-by-value: value 3 of variable \( a \) is passed to the new variable \( n \).

Variable in this case is primitiva data type.

Call-by-reference: object that \( a \) is referred to is passed to the new variable \( n \).

Variable in this case is reference data type.
Modifiers (class, method, variable)

- **Access control modifiers**:
  - default (no modifiers are needed): Visible to the package only.
  - private: Visible to the class only.
  - public: Visible to the world.
  - protected: Visible to the package and all subclasses.

- **Non access modifiers**
  - static: for creating class methods and variables
  - final: for finalizing the implementations of classes, methods, and variables.
  - Abstract: for creating abstract classes and methods.
  - synchronized and volatile: used for threads.
Practice:

- Array
- Classes and Objects
- Call-by-reference and Call-by-value