# LOGIN DETAILS

**ssh** [**plab6056@pe106.comp.nus.edu.sg**](mailto:%20plab6056@pe106.comp.nus.edu.sg)

**ssh** [**userid@hostname.comp.nus.edu.sg**](mailto:userid@hostname.comp.nus.edu.sg)

**elinks index.html**

USERID: plab6056

Hostname: pe106

PASSWORD: 4J8QCuTd

**elinks index.html**

**exit** to exit

# UNIX COMMAND

**mkdir**

**cat**: print out the content in the file

**java Main < test.in**

**jshell Point.java < test.in**

**jshell Point.java**

or

**jshell /open Point.java**

or

**vim test2.in** to open a new file

**:%s/jshell> //gg** to replace jshell to space

**:%s/int/Integer/gc** to replace int to Integer

**cd /mnt/c/users/<USER ID>/Desktop**

**capital y/p** to copy and paste

**dd** to delete the whole line

# LAB 1 NOTES

Reference address

Attributes

**.equals** to compare Strings()

== is comparing memory address

If a and b are pointing to the same thing, change b will also affect a

Need to **@override** this default class to make meaningful comparison

**toString**

by default: ClassName@memoryAddress

pass by **value**: value is copied and passed into method **without modifying the original object**

pass by **reference**: actual address of object is passed into method and **modifies the original object**

**private** cannot be accessed outside of this class

**public** can be accessed by everyone

constructors, getters, setters, setters

**protected** can be accessed by this class and its child class

**static** methods/variables belong to the **class** and **not instances** of an object

# LAB2 NOTES

**String.format()**

**%s** any type

String.format(“Hello %s!”, “World”)

* Hello World!

**%d** double

String.format(“Coordinates: (%d, %d)”, x, y)

**Static**

To declare class level attributes (instead of attributes)

Constant variables

# LAB3 NOTES

If an object is immutable:

Setter methods should return new instance

# LAB4 NOTES

Explicitly declare generic types for static methods

public class Box<T> {

private T item;

public static <T> Box<T> empty() {}

}

Box<? extends T> read from a collection(get)

=>guarantee that whatever extracted from the collection is a subtype of T

Box<? super T> want a collection to consume something(add)

Box<T> if intend to do both

Integer and Double(invariant) are subtypes of Number(supertype)

# LAB5 NOTES

**Functional interface**

Function<Integer, Integer> f=x->x+1

**Exceptions (most specific comes first)**

Try{

} catch(Exception e){

} finally{

}

# LAB6 NOTES

IntStream.range(0, n).forEach(System.out::println)

Source(generate, lazy) -> intermediate(manipulate, lazy) -> terminal(get, eager)

From m to n-1:

Static IntStream range(int m, int n)

From m to n:

Static IntStream rangeClosed(int m, int n)

Arrays.asList(arr).stream()

new ArrayList<>().stream()