Summary Doc

Week 1/2Concepts

Code Reuse: not having to rewrite code, inheritance/encapsulation, objects, overriding

Abstraction: modelling in software the behaviour(methods) and characteristics(attributes) of a real world entity, example Player class, BankAccount class

Encapsulation: encapsulating code in a class, implementation hidden, available through published API/Interface.

True encapsulation: making attributes private…limiting access/visibility

Inheritance (Keywords **extends**, **super**) What is inherited? All methods and none private attributes Is-a relationship Superclass, subclass

Interface: collection of abstract methods/constants (Keyword **implements**..have to write the code) Why is it useful? Don’t have to write the code, when you don’t know how it wil be implemented

Composition/Aggregation : association between classes, data type of an attribute of the composite/aggregate class is a user defined class….’has a’ relationship

method overloading method with different number of or type of parameters

method overriding method which over rides the same method in the class above

Abstract Classes (keyword **abstract**), non instantiable, may/may not contain abstract methods(a method without a body)

Concrete Classes, all the code is included, an instantiable class is a concrete class

Instantiable classes: can created an object of the class using a **constructor**, keyword **new**

declaration versus creation/instantiation

Note: you can declare an object using an abstract class as its data type

UML Diagrams, Hierarchy diagrams, include Interface/Abstract classes, concrete classes

Scope/Visibility (private, public, protected, local variables)

Static (variables/methods)

Arrays…Arrays of objects…populating an array

Enhanced for loop

Ternary operator…..x = (test condition) ? what’s returned if test condition: what’s returned if test condition false

Keyword: final🡪 means a class can not have a subclass, a method can not be overridden, a constant can not have another value.

Naming convention

Clean code

Polymorphism

Dynamic method binding, Method Overriding, Method Overloading

Generic programming

Generic programming (via Inheritance)

JCF

Collection

List

Set

ArrayList

LinkedList

HashSet

TreeSet

Maps

HashMap

TreeMap

Comparable

Comparators

Stacks

Methods/Algorithms/Characteristics/Pros/Cons

Exceptions

Generation/handling

Network programming

Sockets

Client/Server program