/\* START

// X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X //

Java Uses Object-Oriented Programming. Each object has three dimensions. i.e.:

Identity: Human

Attributes: Name, Gender, Height, Weight, Age, Etc.

Behaviour: Walk, Run, Jump, Speak, Sleep, Eat, Etc.

Void: Used when the method does not return anything.

The main method must always be void.

"Public" is an access modifier; meaning it is used to set the level of access.

Access modifiers can be used for classes, attributes, and methods.

TYPE OF CLASSES

public class: The class is acessible by any other class.

default class: The class is accessible only by classes in the same package.

TYPE OF ATTRIBUTES

default: Variable/method is available to any other class in the same package.

public: Accessible from any other class.

protected: Provides the same access as the default access modifier, with the

addition that subclasses can access protected methods and variables of the

superclass.

private: Accessible only within the declared class itself.

Getters and Setters are used to effectively protect data when using classes.

The 'get' method returns the value of the attribute and the 'set' method takes

a parameter and assigns it to the attribute.

Constructors are special methods invoked when an object is created and is used

to initialize them and provide initial values for object attributes.

A constructor name must be the same as its class name and have no return type

Value types are the basic types, and include byte, short, int, long, float,

double, boolean, and char.

MATH CLASS

Math.(abs, ceil, floor, max, min, pow, sqrt, sin, cos, etc.)

abs = absolute, ceil = round down, floor = round up, max = return greater

min = return smaller, pow = power of, sqrt = square root, etc.

i.e. int a = Math.abs(-20)

i.e. int b = Math.abs(Math.min(-6,-20))

When you declare a variable or a method as static, it belongs to the class,

rather than a specific instance. This means that only one instance of a

static member exists, even if you create multiple objects of the class.

It is common practice to use upper case when naming a static variable.

This concept also applies to static methods.

Note: The main method must always be static, but don’t remain it lol.

The 'final' keyword is used to make a variable constant so that it can only

be assigned once (i.e. final double PI = 3.14).

Methods and classes can be marked final so that they cannot be overridden.

Use a wildcard to import all classes in a package.

// X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X //

There are 4 core concepts in OOP:

Encapsulation: The packing of data and methods into a single component.

Inheritance: Class 'X' inherits the members and methods of Class 'Y'.

Polymorphism: A call to a member method will cause a different method to

be executed depending on the type of object that invokes the method.

Abstraction: Focus on essential qualities, rather than specific

characteristics of one particular example.

The idea behind encapsulation is to ensure that implementation details are

not visible to users. This means that the variables of one class will be

hidden from the other classes; and accessible only through methods of the

current class. This is called data hiding.

Encapsulation is achieved by declaring the class' variables as private and

providing public getters and setter methods to modify the variable's value.

Encapsulation provides the following benefits:

Control of the way data is accessed or modified.

More flexible and easily change code.

Ability to change one part of the code without affecting other parts.

Inheritance is the process that enables one class to acquire the properties

(i.e. methods and variables) of another. With inheritance, the information

is placed in a more manageable, hierarchical order.

The class inheriting the properties of another class is the subclass, and the

class whose properties are being inheritied is the superclass.

When one class is inheritied from another class, it inherits all of the

superclass' non-private variables and methods. Remember: private methods

are not inherited from the super class.

Subclass = Dervived class = Child class

Superclass = Base class = Parent class

To inherit from a class, use the keyword 'extend', like so:

"""class Dog extends Animal { // some code }"""

Dog = sub class

Animal = superclass

Polymorphism refers to the idea of "having many forms". This occurs when there is

a hierarchy of classes relted to each other through inheritance.

Briefly, polymorphism is one method, with different implementations.

Polymorphism is also known as method overriding.

Subclasses can override superclasses using the '@Override' method. However, there

are some rules for method overriding. They are:

Both methods should have the same return type and arguments.

The access level cannot be more restrictive than the overridden method's

access level. (i.e. If the superclass is declared public, then the

subclass cannot be private nor protected).

A method declared final or static cannot be overridden.

Constructors cannot be overridden.

When methods have the same name, but different parameters, it is known as

method overloading.

An overloaded method must have a different argument list; the parameters should

differ in their type, number, or both.

Method overloading = Compile-time polymorphism

// X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X //

ABSTRACTION

Data abstraction provides the outside world with only essential information. It

represents essential features without implementing the specifics. Think of a

book. When you hear the term book, you don't know the exact details, such as

page count, color, or size, but you understand the idea, or the abstraction,

of a book. You know it has pages full of information, but not the information

itself.

The concept of abstraction is to focus on the essential qualities, rather than

the specific characteristics of one particular example.

Abstraction is achieved using abstract classes and interfaces.

An abstract class is defined using the 'abstract' keyword:

i.e. """ abstract class Animal { // Code } """

An abstract method is declared with an implementation:

i.e. """ abstract void walk(); """

If a class is declared abstract, it cannot be instantiated. In other words,

you cannot create objects of that type.

To use an abstract class, you have to inherit it from another class.

Any class that contains an abstract method should be defined as abstract.

INTERFACES

An interface is a completely abstract class that contains only abstract methods.

It is defined using the 'interface' keyword.

i.e. """ interface Animal { // Code } """

When working with interfaces...

May contain only static final variables.

Cannot contain a constructor because interfaces cannot be instantiated.

Interfaces can extend other interfaces.

A class can implement any number of interfaces.

Interfaces have the following properties:

An interface is implicitly abstract; you don't need to declare it abstract.

This also applies to methods in an interface.

Methods in an interface are implicitly public.

Note: A class can inherit from just one superclass, but can implemenent multiple

interfaces.

Note: When you implement an interface, you need to override all of its methods.

CASTING

Assigning a value of one type to a variable of another type is known as casting.

i.e. """ int a = (int) 3.14; """

Java supports automatic type casting of integers to floating points, since there

is no loss of value. i.e. "int" ==> "double"

However, casting is mandatory when assigning floating point values to integer

variables. i.e. "double ==> "int" (You lose the decimals).

Note: By default, Java always rounds down. So 2.65 --> 2... And 8.8 --> 8

// X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X //

Java supports nesting classes - a class can be a member of another class. Unlike

a class, an inner class can be private. However, once you declare an inner

class private, it cannot be accessed from an object outside the class.

When you compare objects using the equality testing operator ( == ), it actually

compares the references and not the object values. So while 'int x' may equal

'int y', having two different objects with the same values will return false.

Despire having two objects with the same name, the equality testing returns false,

because we have two different objects (two different references or memory

locations).

ENUMS

An Enum is a special type used to define collections of constants. Enums define

variables that represent members of a fixed set. (i.e. Military rankings)

You should always use ENUMS when a variable can only take one out of a small set

of possible values. If you use ENUMS instead of 'ints' and strings, you

increase compile-time checking and avoid errors from passing invalid

constants, and you document which values are legal to use. Examples of

ENUMS include, but not limited to, month names, days of the week, etc.

API

The Java API is a collection of classes and interfaces that have been written for

you to use. The package(s) can be imported like this:

i.e """ import java.awt.\*; """

The wildcard ( \* ) imports ALL of the classes in the package.

// X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X-X //

END \*/