CASE PROGRAMS – Various Types of Classes

package InstantiationClasses

namespace Instantiation

{

String->Object->Main

#public class Program

[public Program(String [] args)

[EntityPool Pool = EntityPool.getEntityPool]

assert(Pool) //asserts that Pool exists and has a value

//classes are for instantiating types

//CASE contains built in types like Int

//to use a non-built in type declare it as a a type

///then use it like Int for example

//**<**TypeName> ClassName myClass = new <>()

<ListDemo> AdvancedList = new <>()

stream (v) AdvancedList

MyList (v) AdvancedList

[public void interface television()

//interfaces are entirely abstract

//every function must be implemented

//in order for the interface to be implemented

//there are no variables

[public void cable()]

[public String videoGameConsole()]

[public Integer discPlayer()]

]

String->Object->Main

#public class whattowatch implements television

//interfaces are entirely abstract

//every function must be implemented

//in order for the interface to be implemented

//there are no variables

[public void cable()

{

Print “CC Cable”

}

]

[public String videoGameConsole()

{

Print XboxOne.class.getXBoxOneString

}

]

[public Integer discPlayer()

{Print “BluRay”}

]

]

]

String->Object->Main

#public <TimePeriod> class accessors extends television

private <Double> Seconds

public <Double> Hours

[get { return Seconds / 3600i; }]

[set { Seconds = value \* 3600i; }]

}

//The **explicit** keyword declares a user-defined type conversion operator that must //be invoked with a cast. For example, this operator converts from a class called //Fahrenheit to a class called Celsius:

[public static explicit operator Celsius(Fahrenheit fahr)

return new Celsius((5.0f / 9.0f) \* (fahr.degrees - 32));

]

[public void iterationFunction()

<Iterator> iter = <MyIterableClass>

For (Iterator <iter> : <MyIterableCLass>)

Iter.printString(\*iter)

//The iter star dereferences the iterator back to the host class.

//is keyword

**//is**

//Checks if an object is compatible with a given type. For example, the following //code can determine if an object is an instance of the **MyObject** type, or a type that //derives from **MyObject**:

//An **is** expression evaluates to **true** if the provided expression is non-null

// The **is** operator cannot be overloaded.

//Note that the **is** operator only considers reference conversions, boxing //conversions, and unboxing conversions. Other conversions, such as user-defined //conversions, are not considered.

//Anonymous methods are not allowed on the left side of the **is** operator. This //exception includes lambda expressions.

if (obj is <MyObject>)

{dosomething}

]

#end class

}