Loops: if(true) { dothis } else {do this}; while (true){dothis}; do {this} while(ConditionIsTrue); For(int i = 1; I <=100; i++) {dothis}; Foreach (typeIdentifier in arrayName){Dothis}; switch(condition) case1: dothis; break; case2: dothis; break; default: dothis; break; If there is not a default section, no action is taken and control is transferred outside the switch statement. Break causes immediate exit from any statement. ‘Continue’ skips the remaining code in the loop body and proceeds with the next iteration of the loop; Chapter 7: Static variable represents classwide information that is shared by all the objects of the class; Static methods in the same class can call each other directly; A method is able to have optional parameters by placing the parameter to the right of a non optional parameter and assigning it a number; Recursive method is a method that calls itself directly or indirectly through another method; Pass by value – a copy of its value is made and passed to the called function which does not effect the original copy – this stores a copy of the variable; pass by reference – the caller gives the method the ability to access and modify the caller’s original variable – this stores a reference to the variable; Using the key word ‘REF’ to a parameter declaration allows the called method to modify the original variable in the caller; Preceding a parameter with the keyword OUT creates an output parameter – this indicates that the variable will be passed into the called method by reference; Chapter 8: int[] arrayName - it can be reassigned to a new array of a different length; Arrays are reference types because it actually references an array object; Chapter 10: classes and objects – If a member of a class is not declared with an access modifier, it is set to private by default; THIS reference can be used by an object to access a reference of itself and is used when a non-static method is called for a particular object, the method’s body implicitly uses keyword THIS to refer to the objects instance variables and other methods; Data Abstraction – the client cares about what functionality a stack offers, not about how that functionality is implemented; int is an abstract version of integer; Chapter 11: Each derived class constructor must call the base class constructor:base(); Indirect base class is any class above the direct base class in the class hierarchy; Private members are only accessible inside the class itself although a base class’s private members are inherited by the derived class they cannot be directly accessed; every class inherits class object; Constructors are NOT inherited; Questions *LINQ Compare and contrast declarative and imperative programming in LINQ:* var name = from value in tableName where this = this orderby value select value; The Let clause allows us to create a new range variable – from item in items let uppercasestring = item.ToUpper(); *Declarative programming* is when the programmer specifies the conditions that selected elements must satisfy such as in LINQ queries. This expresses *what* the program should accomplish without prescribing *how* to do it in terms of sequences of actions to be taken. *Imperative programming* is when the programmer uses repetitive statements (to filter arrays), which focus on the *steps* required to get the results. Imperative programming is the use of loops and if statements to cycle through the information and sort the necessary elements; *Under What circumstances does the compiler provide a default constructor and how does it initialize instance variables?* The default constructor that takes no parameters is provided when it is invoked if the programmer does not explicitly provide one of their own. Default constructors are invoked whenever an object is instantiated using the **new** operator and no arguments are provided to **new**; *Memory leaks are common in C and C++ by less likely is C#. Why?* The Memory Leaks occurring during garbage collection and destruction process, when the references to the object that manages the resource are lost before the resource can be explicitly released, and it can no longer be released. This is less likely is C# because it has the garbage collector built in and it is used to reclaim memory occupied by objects no longer in use; every object has a special member called a destructor and is invoked by the garbage collector; *How are readonly instace variables different than named constants?* Readonly specifies that an instance variable if an object is not modifiable and that any attempt to modify it after the object is constructed as an error. Consts must be assigned at compile time whereas readonly variables must be assigned at run time. Public readonly int JAKE; Public const int JAKE; *Compare and contrast copy and paste, composition, and inheritance.* Copy and pasting code is a bad way to reuse code because if you change something in one place you must change it everywhere which is time consuming and inefficient. Composition (HAS –A) – a class can have references to objects of other classes as members. Public Class1 FirstName; public TimeClass Oclock; This is the best way to reuse code for smaller projects because it allows you to use the methods and objects of other classes so you do not need to have them in your class file. Inheritance (IS-A) is when the derived class inherits the members from base class (Derived class : base class) in order to use the methods, variables, etc, from the base class. This is a form of polymorphism. This is the best way to reuse code for large projects because it allows you to write general code in the base classes and then use that code inside the derived classes instead of rewriting the same code over and over; *Compare and contrast protected and private access. Which do you prefer and why?* The Private access allows the derived class to inherit the base class’s members but are not able to directly access them by derived-class methods and properties. The protected access allows the children of the base class to use the base class’s members directly so I prefer protected. All non private base class members retain their original access modifier when they become members of the derived class (public, protected); *Discuss the ways in which inheritance promotes software reuse, saves time, and helps prevent errors -Inheritance* allows for polymorphism, which is coding in general so the program can use the code in numerous places. This means the programmer is not typing the same thing in many different files or locations. This saves time by reducing redundancy and prevents errors by reducing the amount of actual typing being done; *What is an abstract class? How would one typically be used in creating a hierarchy?* Abstract keywords indicate that a base class method can be overridden in derived classes. Abstract, in basic terms, means that the class is going to write general code (such as a method without a body) and use a matching method/property from the derived class and use that code to do the work. The abstract classes are usually the parent classes; *Compare and contrast abstract classes with interfaces, which approach would allow completely unrelated classes to share a stucture? -* Abstract classes cannot be used to instantiate objects because they are incomplete. Abstract classes are too general to create real objects. These classes can be used to instantiate objects that are called concrete classes. Interfaces define and standardize the ways in which people and systems can interact with one another. An interface is typically used when unrelated classes need to share common methods so that they can be processed polymorphically. They describe a set of methods that can be called on an object to tell it to perform some task or return some information. Interfaces can only contain abtract methods, properties, indexers and events. (*Public interface IPayable{ })*. Interfaces offer the capability that requires unrelated classes implement a set of common methods. (A method that calculates a payment amount); How does polymorphism enable you to program in the general? Discuss advantages of this - Polymorphism enables us to write applications that process objects that share the same base class in a hierarchy as if they were all objects of the base class. The polymorphism occurs when an application invokes a method through a base class variable at execution (run) time. The key advantages are extensibility: software that invokes polymorphic behavior is independent of the object types to which messages are sent; This allows for less coding, time and mistakes; *A derived class can inherit interface or implementation from a base class. How do inheritance hierarchies designed for inheriting interface differ from those designed for inheriting implementation? -* An interface is typically used when unrelated classes need to share common methods so that they can be processed polymorphically. An interface is often used in place of an abstract class when there is no default implementation to inherit; *What are abstract methods and describe when they are used -* Methods with the keyword, abstract in their declaration must be inside an abstract class. They do not provide implementations (Bodies) and implicitly virtual. The derived class will provide the implementation (if they wish to be concrete classes). This would be used when a method is created in a base class, such as abstract library item, and then implementation is performed in a derived class such as library book; *How does polymorphism promote extensibility? -* Extensibility is when software that invokes polymorphic behavior is independent of the object types to which messages are sent; Polymorphism promotes extensibility: Software that invokes polymorphic behavior is independent of the object types to which messages are sent. New object types that can respond to existing method calls can be incorporated into a system without requiring modification of the base system. Only client code that instantiates new objects must be modified to accommodate new types. This is possible because a subclass object is a super class object as well. When invoking a method from that reference, the type of the actual referenced object, not the type of the reference, determines which method is called. A subclass reference can be aimed at a super class object only if the object is downcast; *Describe the gui controls anchored and docked-* Anchored means that the item is at a fixed distance from the sides of the GUI form. Docked means that it is fixed to the top or bottom of the page and will span the entire length or height of the page; *What is projection and give an example -* Projection is mostly used in LINQ statements. Projection refers to the operation of transforming an object into a new form that often consists only of those properties that will be used. By using projection, you can construct a new type that is built from each object. You can project a property and perform a mathematical function on it. You can also project the original object without changing it; *Preconditions* and *postconditions* allow a programmer to specify **what** a method accomplishes *without* describing **how** the method accomplishes it so if a new developer comes in to make additions or changes to the old code it is easier to understand; *Explain the try –Catch – Finally blocks -* The try catch comes first and contains the code that could possibly throw an exception. The catch block looks for a specific or general exception and then performs some action such as printing a readable statement describing that exception. The catch blocks should be listed in specific or general exception order so by the last catch it will catch anything that may have fallen through. The finally block is guaranteed to execute regardless of whether an exception occurs. It is located after the catch and is great for releasing resources A catch or finally block MUST follow a try block. The finally block is guaranteed to execute regardless of whether an exception occurs. This makes the finally block ideal for code to release resources from the try block. This ensures that even if the program terminates due to an uncaught exception, the resource will be deallocated. TRY {code} catch(typeOfException) { Do this if caught};