Projektarbete Rapport

Product Manager v2.0

**Samlingar**

***Förklara skillnaden mellan Array, List<T> och Dictionary<TKey, TValue>.***

***Array, List<T> och Dictionary<TKey, TValue>*** de alla representerar en samling data.

***Array*** är automatiskt tillgänglig (tillhör System Namespace), som har en definierad storlek med data av samma typ.

Exempel:

int [ ] intArray = new int [10] {1,2,3,4,5,6,7,8,9,10}

intArray [0] = 1;

intArray [9] = 10;

Både ***List<T>*** och ***Dictionary<TKey,TValue>*** tillhör System.Collection.Generic.

***List<T>*** är en av de mest använda generiska samlingsklasserna. Den klassen representerar en starkt skriven lista över objekt som kan nås med hjälp av en index (index börjar från noll.). Kan innehålla element av den angivna typen.

Exempel:

List<T> myList = new List<T>(); // T-data typ

***Dictionary<TKey, TValue>*** - är en ordbok, en tabell med nyckel-/värde par i ingen speciell ordning, еtt bra alternativ till operatören If. Samlingen är dynamisk, kan ändra sin storlek när man lägger till / tar bort ett element. Tillgång till varje element görs med nyckel. Nycklar måste vara unika och vara inte null.

Exempel:

Dictionary <TKey,TValue> myDictionary =new Dictionary <TKey,TValue>();

***När skulle man välja en Dictionary<TKey, TValue> framför List<T>?***

***Dictionary <TKey,TValue>*** är den bäste, när varje objekt i listan garanterat har en unik nyckel. Men det tillåter inte att samma nyckel används två gånger, och det lägger ingen ordning - man kan inte iterera över ordboken "i ordning" eftersom det finns ingen ordning.

***Finns det något tillfälle då vi skulle välja en Array framför en List<T>?***

Om man vet att data har en fast storlek(längd) och man vill mikrooptimera av någon mycket specifik anledning, kan en array vara användbar***.*** Och Array gör en hel del bit-shifting, så en byte [ ] är ganska mycket viktig för kodning.

**Objektorienterad programmering**

***Vad innebär inkapsling och hur uppnår vi det?***

***Inkapsling*** är en av de huvudfunktionerna i objektorienterad programmering. Det är en mekanism, som vi använder för att binda samman data och funktioner till en enda enhet klass). I inkapsling kommer klassens variabler att vara dolda från andra klasser och kan endast nås genom metoderna för deras nuvarande klass. Inkapsling döljer implementeringen av ett objekt från slutanvändaren, vilket i C# utförs med hjälp av åtkomstmodifierare (privat, public, protected, internal).

Det kan uppnås genom att deklarera alla variabler i klassen som privata och använda C#-egenskaper(properties) i klassen till set och get värden av variabler.

***Förklara de 4 grundpelarna i objektorienterad programmering.***

***Abstraktion*** - är något som inte har en specifik representation (instans) i verkligheten och förenar i sig en viss gruppegenskaper för objekt kopplade till en abstrakt grupp (till exempel träd: detta ord beskriver objekt som har vissa egenskaper som vi säkert kan specificera: antal grenar, namn osv. Dock själva instansen det finns inget träd, eftersom det är en abstrakt uppsättning egenskaper som är inneboende i olika klasser, som till exempel björk, tall, ek).

***Inkapsling*** – är en mekanism för att binda samman data och funktioner till en klass.

***Arv(inheritance)*** - det är en process som tillåter en klass (underklass) att utvidga en annan klass (basklass) genom att ärva dess funktionalitet (metoder).

class

**Object**

class

**Shape**

class

Shape

**Square**

***Polymorphism*** – härstammar från grekiskan (poly och morphs) och betyder ”mångformig”. Process som ger en förmåga för klasser att implementera olika metoder med samma namn. Polymorphism ger också en möjlighet att anropa en härledd klass metoder genom basklassreferens under körning baserat på våra krav. Det finns två olika typer av polymorphism: statisk (compile time) och dynamisk(run time). Statiska typen implementeras via overloading metod. Dynamiska typen(runtime) händer när applikationen kör. Denna typ kan uppnås med två olika tekniker: arv och interfaces.

Fordon

Typ

Tvåhjuling

Fyrahjuling

Trehjuling

**Förklara vilka olika typer av relationer mellan klasser som kan finnas i objektorienterad programmering.**

***Inheritance( )* –** är **IS A** typ av relationer, som tillåter en klass att förvärva egenskaperna och beteendet hos en befintlig klass.

class Animal{

//kod

}

//Cat ärver från Animal

class Cat:Animal {

//metoder av Animal

//metoder av Cat

}

***Association( )*** – är **HAS A** typ av relationer där objekt av en typ på något sätt är relaterade till objekt av en annan typ. Till exempel, ett objekt av en typ innehåller eller använder ett objekt av en annan typ. Association kan vara 1 till 1, 1 till många, många till 1 och många till många. Till exempel spelar en spelare för ett specifikt lag:

class Team{//kod

}

class Player{

public Team Team {get; set; }

}

class Team

Attributes

Operations

+Team()

class Player

Attributes

+Team:Team

Operations

+Player()

Player Team

1

\*

***Aggregation*(**  **)** – är **HAS A** typ av relationer, en enkelriktad association. Detta förhållande existerar när en klass äger men delar en referens till objekt från en annan klass.

class Address{

//kod här

}

class StudentClass{

private Address studentAddress;

//kod här

}

***Composition ( )*** - är **PART OF** typ av relationer, en stark typ av association. Den händer när en klass äger och innehåller objekt från en annan klass.

Room

Part of

House

class Room { //här är kod

}

class House{

private Room room;

// här är kod

}

**Versionshantering**

***Varför vill vi versionshantera vår kod?***

Det hjälper oss att hantera ändringar av koden över tid, som håller reda på alla ändringar som görs i koden. När en bugg hittas kan vi gå tillbaka och jämföra med tidigare versioner av koden för att fixa buggar, vilket minimerar problem för alla, som arbetar med koden. Det gör möjligt för all utvecklingslag att jobba snabbare och mer effektivt.

***Hur skapar man ett repo?***

Man kan göra det genom terminal, GitHub eller genom verktygsfältet (Git->Create GitRepository).

Genom GitHub: Create New repository.

Genom Terminal:

git init

***Hur skapar man en commit?***

Team Explorer🡪Home knappen 🡪Git Changes🡪message🡪Sync🡪Push

eller genom terminal:

git commit

git commit -m ”ditt commit meddelande”

***Hur kan man se hur kodbasen (projektet/solution) såg ut vid en specifik kommit?***  git log -p - (commit nummer 1-är den sista)

***Hur kan man se vilka ändringar gjordes i en specifik commit?***

git show <commit namn>

***Vad är GitHub?***

GitHub är ett socialt nätverk, en plattform för att lagra, dela och hantera källkod som låter dig och andra arbeta tillsammans med samma projekt.

***Hur får man upp sin kod till GitHub?***

git push

**Testdriven utveckling**

***Vad är testdriven utveckling?***

Det är en process som använder tester för att utveckla din applikation. Målet med att skriva test är att se till att koden du skriver fungerar som förväntat och att du inte bryter något när du lägger till nya funktioner; minimera antalet fel i programmets arbete, spara tid och uppnå en ”renare” kod.

***Var innebär det att jobba testdrivet?***

Det gör dig till en mer kompetent utvecklare.