Inlämningsuppgift 3

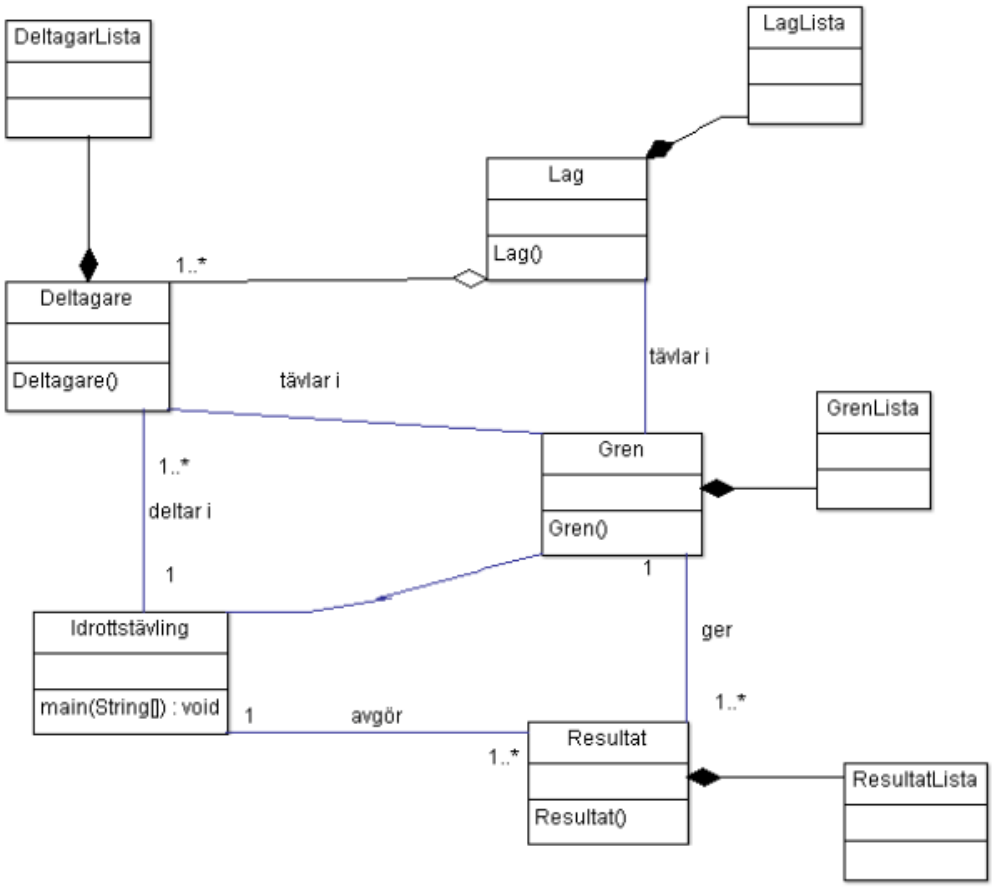
Grupp: 30
Simon Schwieler (SiSc7379)
Robert Dighem (RoDi0231)
Aron Holmquist (ArHo2993)

Objektorienterad programmering

Höstterminen 2015

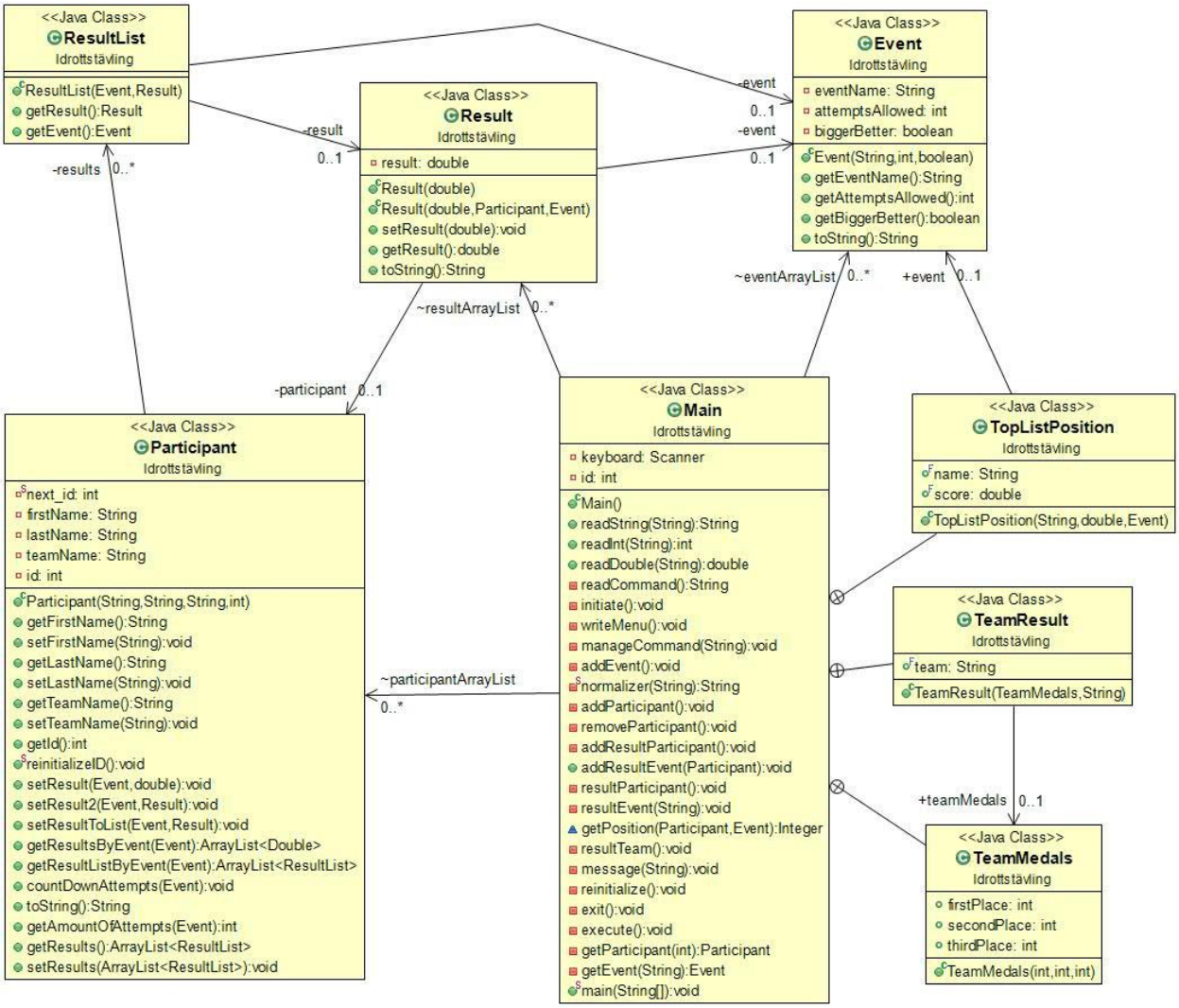
Kursansvarig: Henrik Bergström

1 Design



Det ovan är vårt första utkast på ett klassdiagram. Här har vi klasser för varje lista men det kom vi att ändra på senare.

Vi har utgått från programskelettet i vår utformning av main-klassen. Vi har valt att dela upp klasserna i Main, Participant, Result, Event och Resultlist i vårt färdiga program. Vi valde att ha en switch-sats precis som i programskelettet. Från början hade vi en egen klass för Lag(Team) men tog sedan bort den för att istället anropa den genom Participant(Deltagare).


<<Java Class>>
ResultList
Idrottstävling
ResultList(Event,Result)
getResult():Result
getEvent():Event
-results
0..*
-result
0..1
<<Java Class>>
Result
Idrottstävling
result: double
Result(double)
Result(double,Participant,Event)
setResult(double):void
getResult():double
toString():String
-event
0..1
-event
0..1
<<Java Class>>
Event
Idrottstävling
eventName: String
attemptsAllowed: int
biggerBetter: boolean
Event(String,int,boolean)
getEventName():String
getAttemptsAllowed():int
getBiggerBetter():boolean
toString():String
~resultArrayList
0..*
~eventArrayList
0..*
+event
0..1
-participant
0..1
<<Java Class>>
Participant
Idrottstävling
next_id: int
firstName: String
lastName: String
teamName: String
id: int
Participant(String,String,String,int)
getFirstName():String
setFirstName(String):void
getLastName():String
setLastName(String):void
getTeamName():String
setTeamName(String):void
getId():int
reinitializeID():void
setResult(Event,double):void
setResult2(Event,Result):void
setResultToList(Event,Result):void
getResultsByEvent(Event):ArrayList<Double>
getResultListByEvent(Event):ArrayList<ResultList>
countDownAttempts(Event):void
toString():String
getAmountOfAttempts(Event):int
getResults():ArrayList<ResultList>
setResults(ArrayList<ResultList>):void
~participantArrayList
0..*
<<Java Class>>
Main
Idrottstävling
keyboard: Scanner
id: int
Main()
readString(String):String
readInt(String):int
readDouble(String):double
readCommand():String
initiate():void
writeMenu():void
manageCommand(String):void
addEvent():void
normalizer(String):String
addParticipant():void
removeParticipant():void
addResultParticipant():void
addResultEvent(Participant):void
resultParticipant():void
resultEvent(String):void
getPosition(Participant,Event):Integer
resultTeam():void
message(String):void
reinitialize():void
exit():void
execute():void
getParticipant(int):Participant
getEvent(String):Event
main(String[]):void
<<Java Class>>
TopListPosition
Idrottstävling
name: String
score: double
TopListPosition(String,double,Event)
<<Java Class>>
TeamResult
Idrottstävling
team: String
TeamResult(TeamMedals,String)
+teamMedals
0..1
<<Java Class>>
TeamMedals
Idrottstävling
firstPlace: int
secondPlace: int
thirdPlace: int
TeamMedals(int,int,int)

2 Funktionen lägg till resultat

Vi börjar med att be användaren att skriva in vilket ID resultatet ska läggas in för, om ID:t inte finns i programmet så går programmet tillbaks till huvudmenyn. Om ID:T finns inlagt i systemet så anropas en ny metod som ber användaren om grennamn om grenen finns inlagt i programmet så kommer deltagarens förnamn, efternamn och grenens namn skrivs ut samt en förfrågan om resultatet. När detta gjorts så läggs resultatet in i resultList som är listan vi använt os av för att koppla ihop deltagare med resultat och gren.

Vi inleder med att loopa igenom vår deltagarlista för att få fram rätt objekt av deltagare. Om rätt deltagare hittats så anropas metoden addResultEvent() och skickar med participant p som vi fick ut av loopen. Sedan tar vi in en sträng som vi jämför mot metoden getEvent() där vi loopar igenom eventArrayList för att se om det finns ett event med det namnet. När deltagare och ett event har hittats så görs några kontroller innan ett nytt resultatobjekt skapas sedan anropas metoden setResultToList som lägger in objektet i en arraylist.

3 Funktonen resultatlista för lagen

Först kollar vi om vår arraylist för participant är tom, är den det får man felmeddelande och går ut ur metoden.

Annars loopar vi igenom participantArrayList och eventArrayList och anropar en metod som heter getPosition och skickar med participant och event. Denna metod skapar en arraylist för toppositionerna där vi lägger in objekten som vi skickat med. Denna lista sorteras sen efter eventets biggerbetter värde i en collections.sort metod.

I slutet av metoden så har vi en for-loop som sköter utskrivningen av placeringssiffror. När vi sedan har fått reda på participantens placering så kollar vi om positionen inte är lika med null. Vi tar fram participantens lagnamn och sätter det lika med teamMedals. Därefter kollar vi positionen och adderar medaljer utifrån placering. Sedan har vi en comparemetod som jämför placeringar för utskrift. I comparemetoden tar vi teamresults som argument och jämför dom utifrån första, andra och tredje placering.

4 Normalisering av namn

Vi har gjort en metod som heter normalizer, den använder vi vid de tillfällen där strängar ska normaliseras istället för att upprepa koden vid varje tillfälle.

Det första vi gör i metoden är att ta bort alla whitespaces genom .trim när man skriver in eventet, därefter gör vi om strängen till lowercase. Vi sätter därefter in eventname i en array och tar nollte index och sätter det till uppercase och returnerar den nya strängen.

5 Arrayer och ArrayList

En array har vi använt en gång i vår normaliseringsmetod, ArrayList har vi ett flertal av. Vi har arrayList för participant, event och result. Dessa tar in objekt från tillhörande klasser. Utöver dessa arrayLists så har vi skapat en arrayList för toplist, detta för att enkelt kunna spara och få fram toppositionerna.

Vi valde att skapa arrayList för just dessa eftersom att det är ett enkelt sätt att samla mängder på och sedan jämföra eller koppla ihop dom.

6 Statiska variabler och metoder

Vi har valt att göra vår normaliseringsmetod statisk för att den inte påverkar klassens tillstånd. På samma sätt har vi gjort en statisk metod som används vid reinitialisering för att nollställa vår counter för deltagarID, även den tillhörande variabeln är statisk. Dessa kommer inte heller påverka objektets tillstånd.

7 Reflektion

Det har varit en ansträngande uppgift och ofta har det varit väldigt motigt att arbeta med den. Men när bitarna faller på plats är det självklart kul! Nivån har varit väldigt hög för en nybörjarkurs, och vi skulle gärna sett att det fanns ytterligare en föreläsning eller någon form av labb för att komma igång snabbare. Tidsmässigt så har det tagit mycket längre tid än vad vi väntade oss, men vi hade antagligen kunnat lägga upp en tidsplan som hade förenklat delmålen (ex att Add Participant ska vara klar inom ett visst datum etc). På det stora hela känner vi att vi lärt oss mycket av uppgiften trots att den har avskräckt oss från att röra programmering stundtals! Handledningstillfällena har ofta varit väldigt uppbokade och det kanske skulle behövas ytterligare lärare eller fler tider. När det gäller designen har vi tänkt på att bryta ut saker i metoder för att inte upprepa för mycket kod. Det är något som förenklar programmeringen oerhört. Namngivningen är svår när det blir många metoder som gör snarlika saker, men jag tyckte ändå att vi skötte det snyggt, och det finns en god koppling mellan namngivningen och metodernas syfte, likaså variablernas namn. Detta är ändå något vi alla känner att vi kommer bli bättre på ju mer vi programmerar. Skyddsnivåer försökte vi efterlikna det som sägs under föreläsningarna (instansvariabler private, metoder som ska användas utifrån public etc). Detta fick vi god rutin på och tillslut så kommer det av sig själv. Innan vi satte igång hade vi kunnat gå igenom det viktiga i kursen ytterligare en gång, då vi ibland fastnade på saker som sades under de tidigare föreläsningarna och som vi då hade glömt. Så en tillbakablick på tidigare sagda saker hade helt klart underlättat, samt en tidsplan som nämndes ovan!

Källförteckning

[1] J. Lewis och W. Loftus, Java Software Solutions, sjunde upplagan red., Pearson Education, 2012.