Teknisk Dokumentation

Team06 EDAF45

20 februari 2017

1 Utvecklingskommandon

För att bygga projektet och generera .jar-filen:

- Befinn dig i **/pvgvt17-team06-production/dasUberEnduroApplicationen* i terminalen.
- Skriv på kommandoraden "./gradlew build".

För att köra acceptanstesterna:

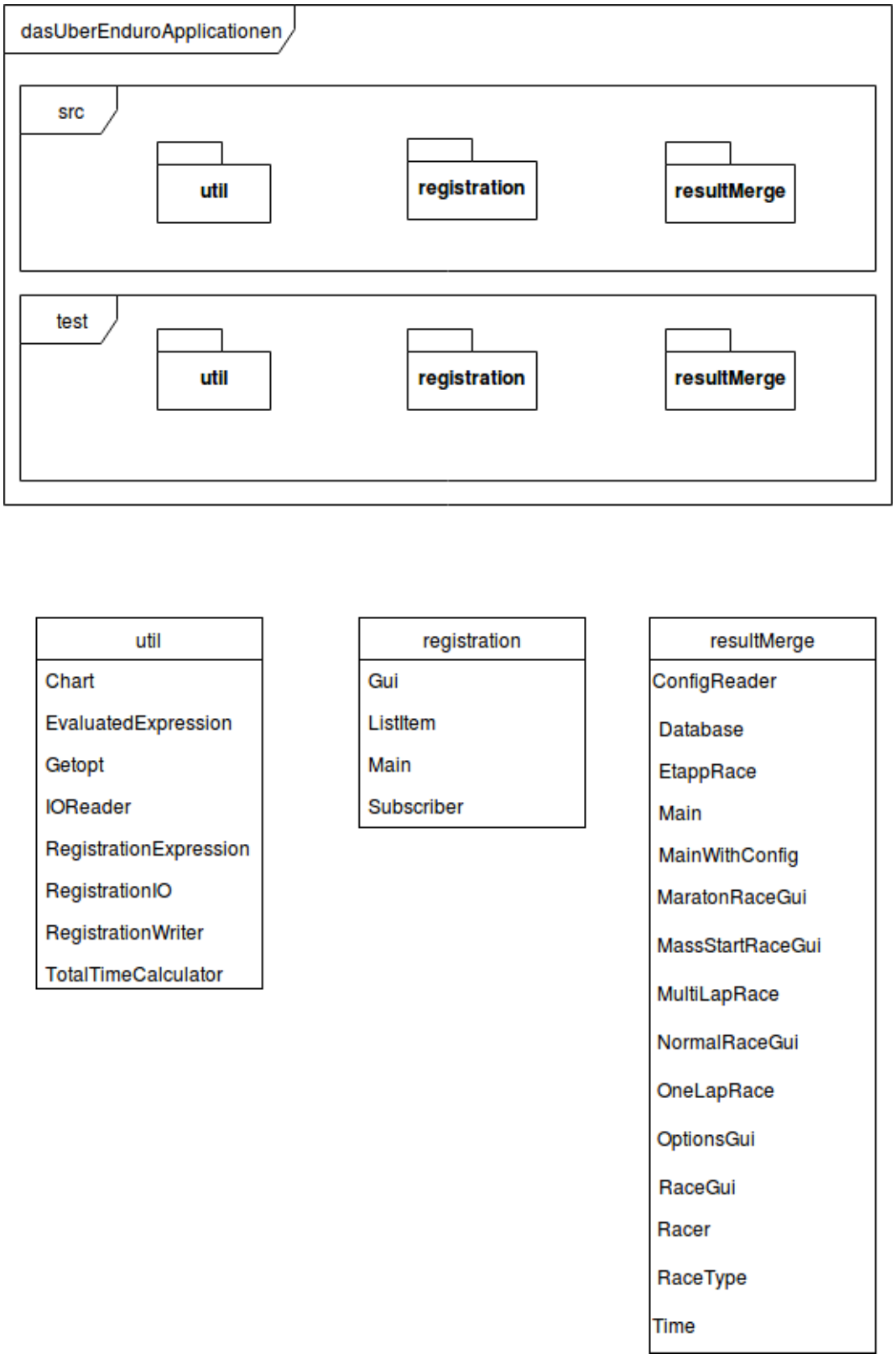
- Befinn dig i **/pvgvt17-team06-production/* i terminalen.
- Skriv på kommandoraden "./runAcT.sh"

2 Designbeskrivning

Vårt program DasUberEnduroApplicationen består av 3 paket, util, registration och resultMerge. Figur 1 visar alla paket samt listor av alla klasser i src. Varje Racer är ett eget objekt som håller koll på information om sig själv som: namn, startnummer, tävlingsklass och övrig information. Olika sorters tävlingar beskrivs av interfacet RaceType, som implementeras av OneLapRace och MultiLapRace. Vi använder oss av Strategy-mönstret som gör att varje Racer-objekt har ett RaceType-objekt i sig som bestämmer vilken typ av tävlingssort det Racer-objektet ska tävla i. RaceType-klasserna har ansvar för att generera sista delen av resultatsträngen för varje Racer.

Exempel, MultiLapRace: Om RaceType är ett MultiLapRace skriver först Racer-objektet ut de två första raderna i strängen som är StartNummer; Namn". Därefter anropas RaceType.genResult() som genererar information från just den tävlingssorten. I detta fallet ser fortsättningen av strängen ut så här: #Varv; TotalTid; Varv1; Varv2; Varv3; Start; Varvning1; Varvning2; Mål.
Utöver att generera strängen håller även RaceType reda på tider som hör till denna tävlingssorten.
Databasen är sedan en samling av alla Racer-objekt, där dem sparas i en HashMap med dess startnummer som nyckel. i Klassen ResultWriter i util-paketet ska resultatet skrivas ut på en textfil. Den genererar en header beroende på vilken tävlingssort det är och itererar sedan över databasen med Racers och kallar

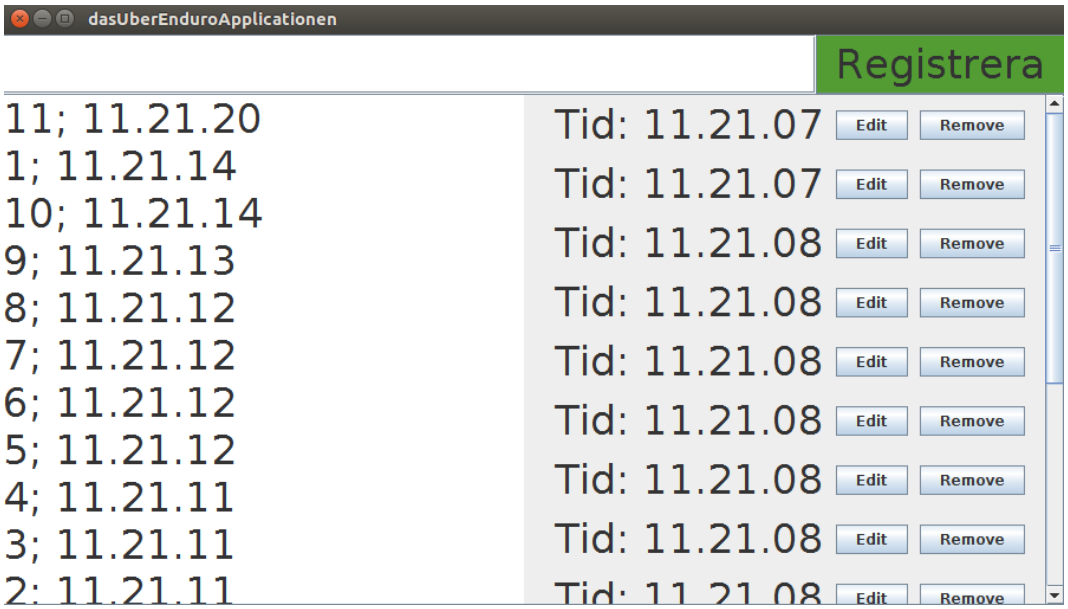
sedan på varje Racers toString()-metod. IOReader-klassen i util-paketet står för inläsning av textfiler. För att programmet ska fungera behövs en namnfil, en starttidsfil och en sluttidsfil. Dessa måste vara i textformat och formatterade precist efter en viss standard. Detta för att vår IOReader ska kunna läsa korrekt.
Racer-klassen implementerar interfacet Comparable, och jämför sig själv med andra Racers först efter antalet varv och sedan efter totaltid. Med denna lösning kan vi använda den i både varvlopp och maratonlopp.



Figur 1: Paketstruktur och listor av klasser i programmet dasUberEnduroApplicationen.

3 Gui

Figur 2 visar hur användargränssnittet ser ut för vårt program. I fältet högst upp skriver man in startnummer för varje Racer. När man klickar på Registrera hamnar den racern i kolumnen till vänster och tilldelas en starttid. Om man inte kan se startnumret klickar man bara på Registrera utan att ange startnummer, då hamnar starttiden i den högra kolumnen tillsammans med två knappar: Edit och Remove. Edit-knappen används om man vill skriva in ett korrekt startnummer, och isåfall hamnar den sedan i den vänstra kolumnen på samma sätt som vid initial registrering. Om man istället klickar på Remove-knappen försvinner starttiden från registeringsprogrammet.



Figur 2: Gui av dasUberEnduroApplicationen.

4 UML-diagram

4.1 Gui

4.2 ResultMerge

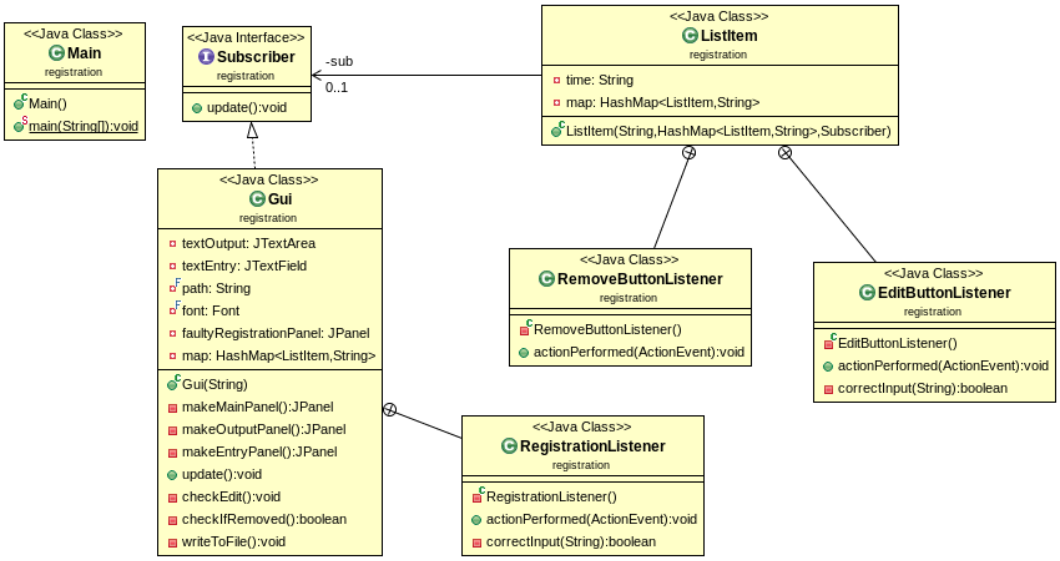
4.3 Util

4.4 Flow

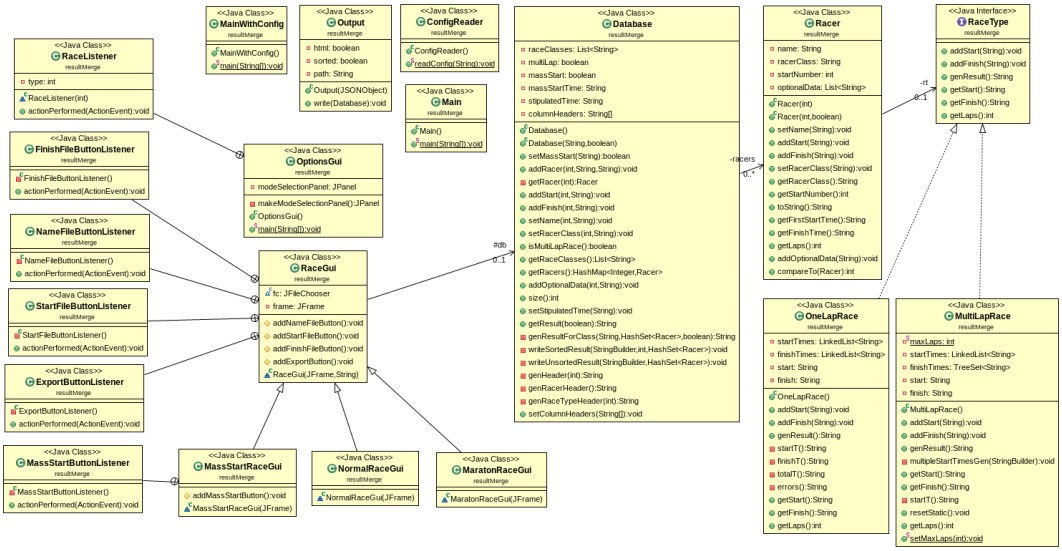
5 Framtida projektplaner (Work in progress)

Vi arbetar på ett antal parallella områden som påverkar UML-diagramen mycket. Nedan är de viktigaste.

- **Etapplopp** - Vi har börjat implementera logiken för etapplopp, men det saknas helt konfigurationsstöd. Planen är att detta ska följa med i Release 2.



Figur 3: UML av GUI-paketet.

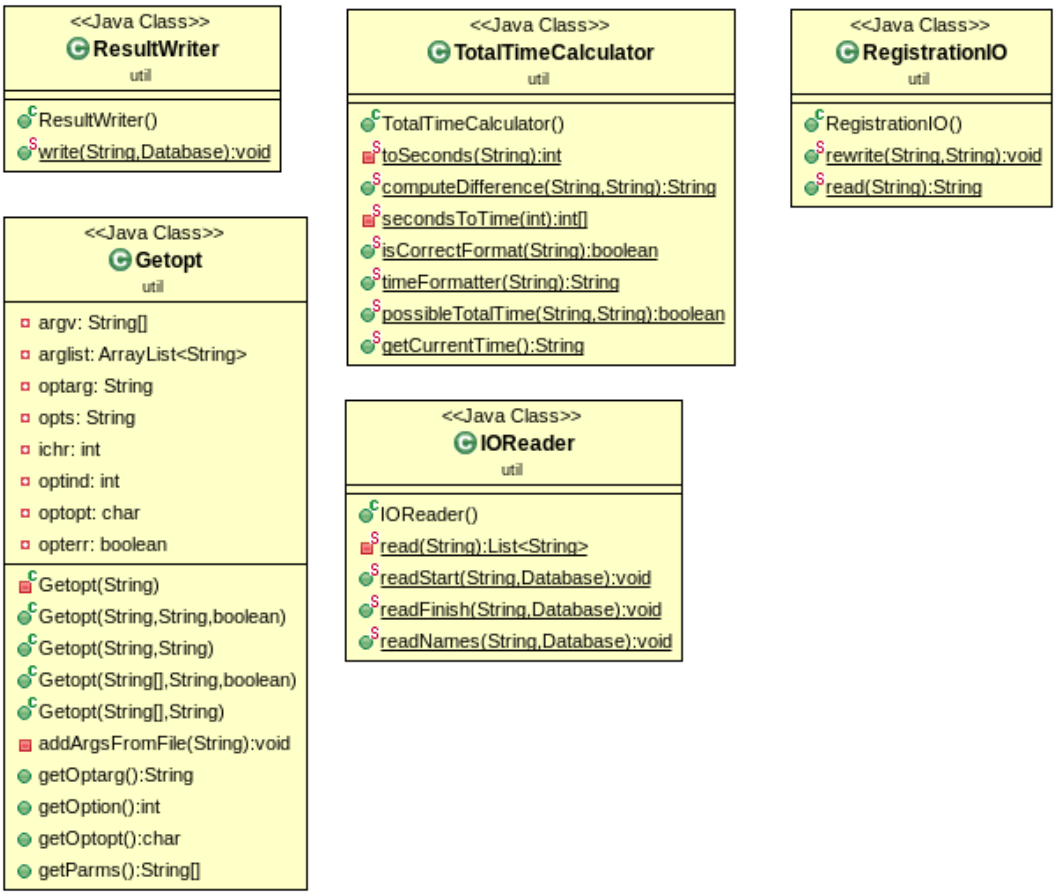


Figur 4: UML av ResultMerge-paketet.

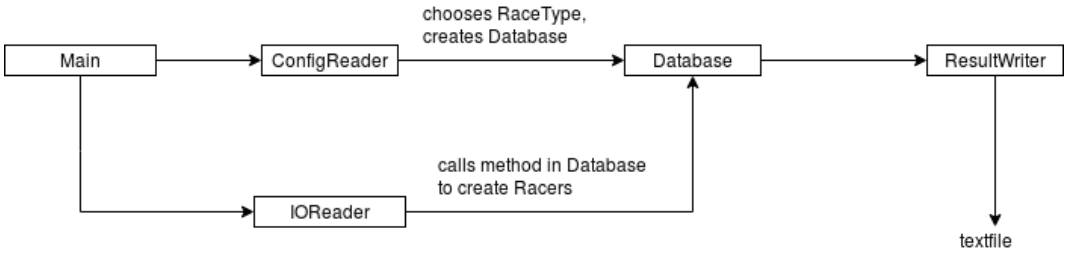
- **Main** - För tillfället håller vi på att byta hur programmet startas, från parametrar till en konfigurationsfil. Detta är i princip klart men inte stabilt ännu.

- **Mer generella tider** - För att etapplopp ska kunna fungera måste vi kunna skicka mer information än bara tiden. Vi har därför börjat implementera denna struktur, men snart krävs en stor refaktorisering eftersom båda sätten att representera tider används för tillfället i de respektive RaceType-klasserna.

- **Util** - Vi håller på att utökar vårt Util-paket för att formatteringen av strängar ska ske på samma ställe. Detta kommer ändra en hel del beroenden och duplicerad kod i stort sätt hela projektet.



Figur 5: UML av Util-paketet.



Figur 6: Programflöde från inläsning till utskrift.