# Technical Documentation

Enduro Race System

*Team 3*

February 27, 2017

# Contents

# 1 Inledning

Detta dokument är skrivet för att möjliggöra vidareutveckling, underhåll och testning av tidtagningssystemet Enduro Race System .

# 2 Bakgrund

På begäran av kund drivs utvecklingen av Enduro Race System som är ett tidtagningssystem främst ämnat, men inte begränsat till, tidtagning och resultatgenerering för tävlingar i motorsporten enduro.

Hanterade tävlingsformer är:

- Maratonlopp
- Varvlopp

# 3 Systeminformation

Systemet samlar in tidsregistreringar för deltagare vid olika stationer. Dessa registreringar sammanställs när alla tävlande har kommit i mål, och tidsregistreringarna länkas mot förregistrerade deltagare, varvid en resultatlista genereras. Systemet består av två subsystem: registrering och resultatgenerering.

Data för generering av resultat mellanlagras i textfiler. Även resultatet presenteras som en textfil (.txt) i "CSV-format" för inläsning till MS Excel, samt i "HTML-format" för webpublicering. Se appendix A för exakt filformat.

Huvudsystemet är uppdelat i två sub-system:

- Registreringssystem
- Resultatsystem

Dessa subsystem och deras interaktioner beskrivs i denna avdelning.

## 3.1 System: Registrering

Registreringssystemet ligger i paketet `registration` och innehåller klasserna `Registration` och `RegistrationProgram`. Det grafiska användargränssnittet är utvecklat i mjukvaruplattformen JavaFX. Nedan följer en mer detaljerad genomgång av de individuella klassernas funktionalitet och samspel.

### 3.1.1 Uppgifter, använding och ansvar

Registreringssystemet är ett separat program som körs parallellt på flera stationer av många användare samtidigt, utan att deras interaktion påverkar varandra.

Registreringssytemets ansvar är att samla in data i form av tidsregistreringar länkade till ett startnummer. Operatören väljer om det är en start/mål(varv)-station vid uppstart av programmet. Operatören av programmet skriver manuellt in ett startnummer varvid en tid registreras. Datan sparas i en fil som manuellt (via USB) överförs till resultatsystemet. Se appendix A för filformat.

Registreringssystemet stöder också "förregistrering" av förare, d.v.s. att ett startnummer i efterhand adderas till en tidtagning. För att förhindra att registreringar förloras vid eventuella system-krascher sparas denna tidsregistrering i en separat fil som heter `preregtimes.txt`.

### 3.1.2 Klassbeskrivning, UML, och designmönster

Övergripande designmönster för hela detta paket är MVC. Där modellen är `RegistrationProgram` och `Registration` klasserna. View består av `RegistrationApplication` samt resten av filerna i `view` paketet. Controller består av alla klasser som finns i `controller` paketet.

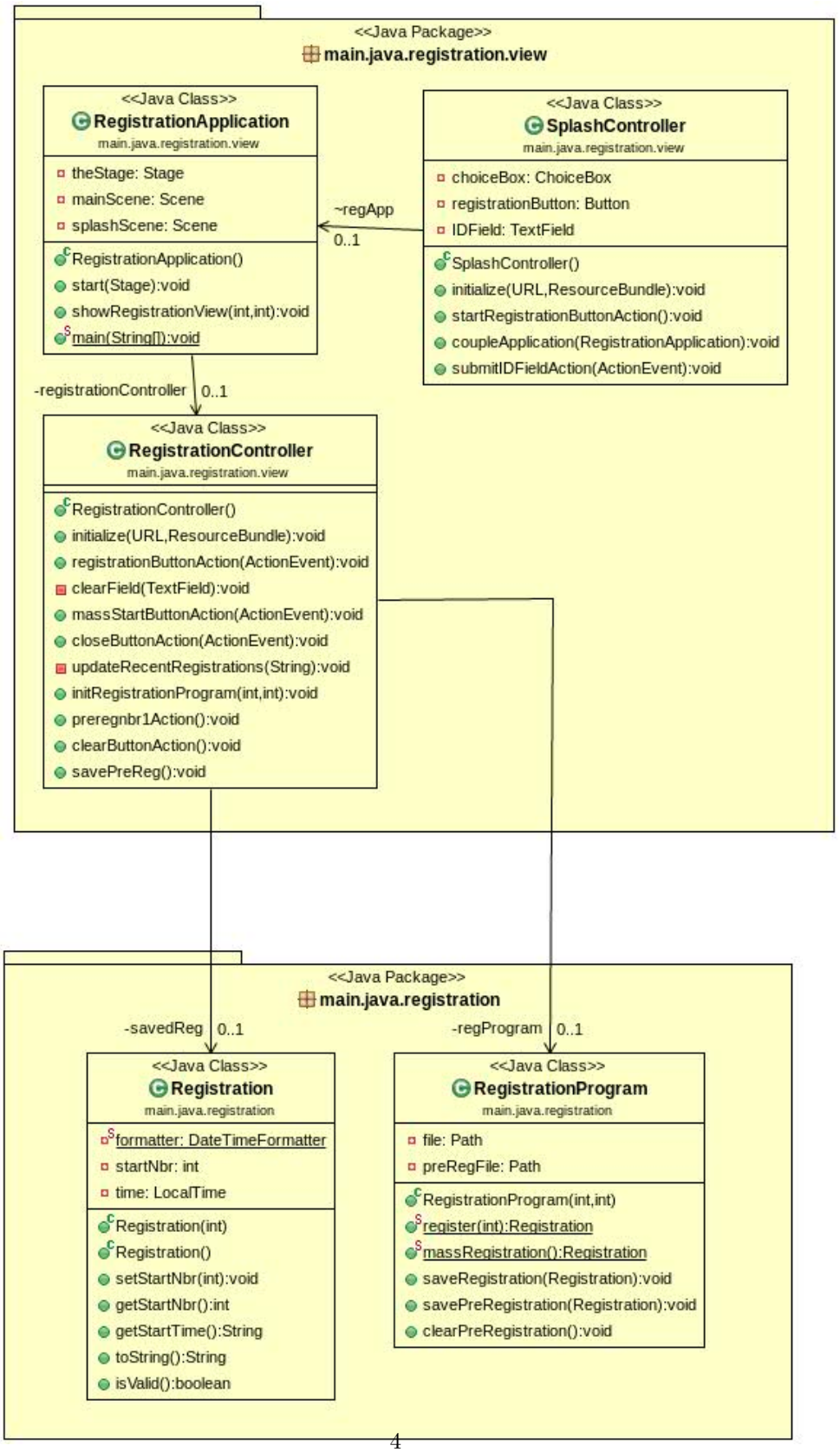




Figure 1: UML för `Registration`-paketet.

`RegistrationProgram` är en klass som ansvarar för att skriva insamlad data till textfiler. Kontruktorns inparameterar är ett heltal som identifierar den station som kör applikationen, samt ett heltal som talar om vad det är för registrering som skall utföras. Detta heltal avgör sedan till vilken fil insamlad data skall skrivas. Registreringar sparas genom att skapa Registration-objekt. Programmet stödjer även funktioner som masstart där alla tävlande startar med samma tid, samt förregistrering. `RegistrationProgram` har ett starkt beroende till klassen `Registration`.

`Registration` är en klass som representerar de individuella registreringarna och ansvarar för att lagra ett startnummer samt tiden då objektet skapades. Klassen ansvarar också för att tidsformatet ska vara analogt för alla registreringar i alla textfiler. Det är toString-metoden i denna klass som skriver data till filer.

`RegistrationApplication` ligger i paketet `view`. Denna klass innehåller main-metoden för att initiera och starta igång GUI:t. Det finns även ytterliggare ett paket som heter `controller` där finns klasser som hanterar kopplingen mellan programmet och användargränssnittet.

### 3.1.3 Output-filer

Filerna som skapas utav `RegistrationProgram` sparas i mappen "./registered_times". Dessa filerna innehåller start eller slut-tider beroende på vilken typ av station det är. Förregistrerade tider sparas också i en separat fil i rotmappen. Se appendix A för exempel.

## 3.2 System: Resultat

Resultatsystemet ligger i paketet `sorter` och innehåller ett flertal klasser som ansvarar för lagring av data, inläsning av data, sortering samt generering av resultatfil.

Nedan följer en mer detaljerad genomgång av funktionalitet samt samspel av klasserna i resultatsystemet.

### 3.2.1 Uppgifter, använding och ansvar

Generera resultat med hjälp av resultatfilen:

1. Läsa in inställningar från konfigurationsfil.
2. Registrera användare utifrån förbestämd participant fil.
3. Länka tidsregistreringar till användarna.
4. Sortera enligt bästa tid.
5. Generera en resultatfil.

### 3.2.2 Klassbeskrivning, UML, och designmönster

I figur 1 visas UML för delar av `ResultatGenerator`-paketet.

Ett generellt data-flöde beskrivs här:

Centralt finns en `ParticipantList` som innehåller insamlad data för varje deltagare. `ResultGenerator`-klassen adderar data till denna genom att anropa instanser av `Reader`. `Reader` läser textfiler som lägger till data i `Participant`-objekten i `ParticipantList`. `ResultGenerator` skapar en resultfil när samtliga filer har lästs in.

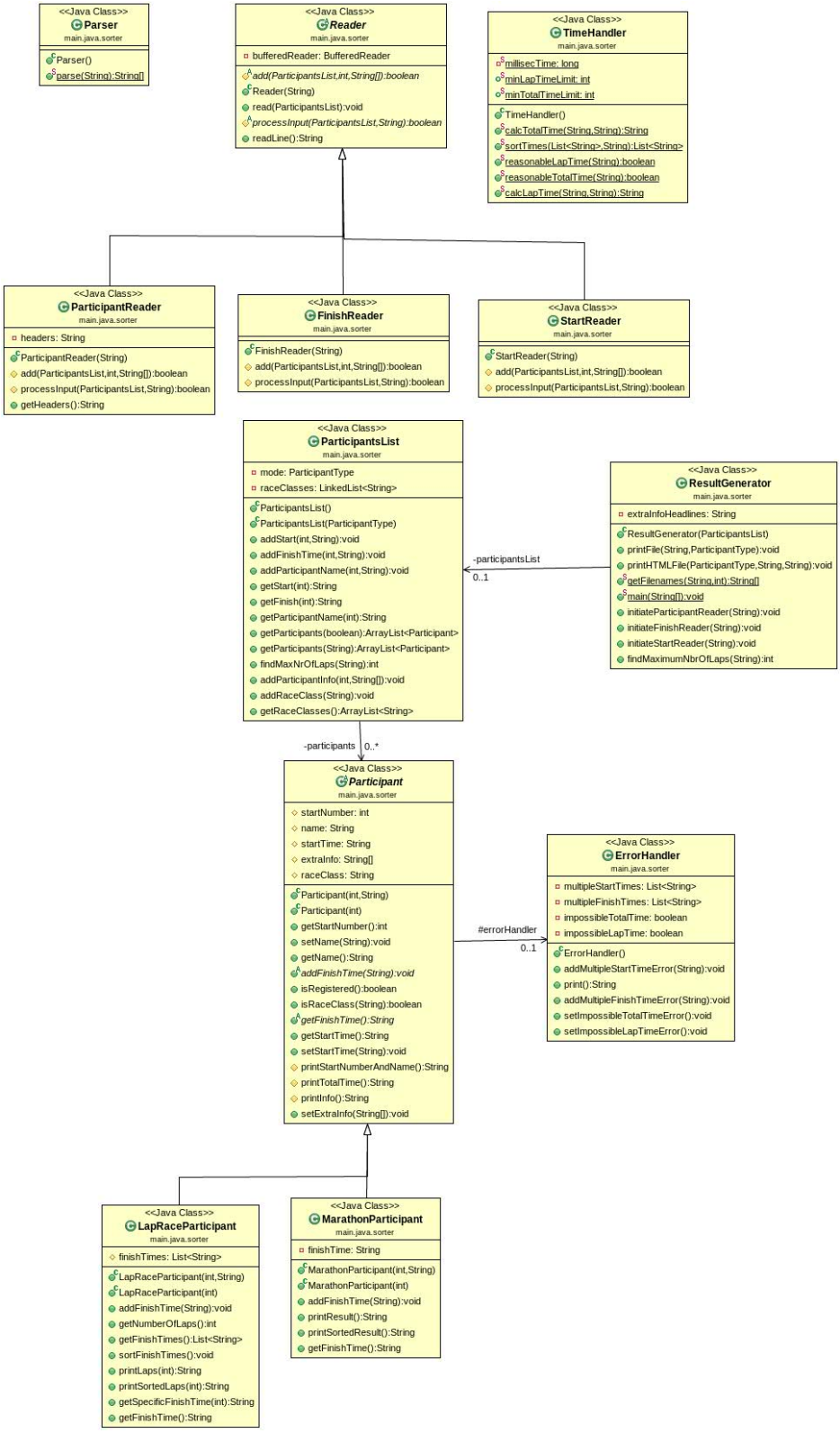


Figure 2: UML för delar av `ResultGenerator`-paketet.

En beskrivning av centrala klasser ges här:

`ResultGenerator` har main-metoden, läser in konfigurationsfilen, ställer in programmet efter önskade inställningar och bygger `ParticipantList` m.h.a. `Reader` och skapar sedan en resultatfil.

`ParticipantList` innehåller `Participant`-objekt, och fungerar som ett gränssnitt för att få tillgång till deltagardata.

`Participant` är en abstrakt superklass som representerar en deltagare i tävlingen. Denna klass ansvarar för att lagra den tävlandes data i form av bland annat startnummer, startid samt sluttid:er(varvtider). Data läses in från textfiler som genereras av registreringssystemet.

`MarathonParticipant`-subklassen representerar det traditionella marathonloppet där varje deltagare har en starttid och en sluttid förutsatt att deltagaren går i mål. För att kunna hantera felaktig indata måste klassen kunna lagra flera starttider samt måltider.

`LapRaceParticipant`-subklassen representerar varvloppet, vilket kan ses som en sekvens av marathonlopp där tävlingen pågår under en bestämd tid. Under denna tid vinner den deltagare som lyckas springa flest varv. Detta ställer nya krav på deltagarklassen som måste lagra starttid, varvtider samt sluttid.

`Reader` är den klass som bygger `ParticipantList` genom att läsa från inputfiler. Klassen följer template-mönstret och är abstrakt för att hantera olika sorters input-filer motsvarande olika typer av lopp.

`TimeHandler` (syns ej i UML) sköter beräkningar som berör tider. Ansvarar för att:

- Beräkna totaltiden då indata ges av en starttid och sluttid.
- Utvärdera om den beräknade totaltiden är rimlig för det givna loppet.
- Sortera tiderna efter kortaste till längsta i en lista.

### 3.2.3 Input-filer

Input-filer kan preciseras i konfigurationsfilen om så önskas. Om de oförändrade värdena i `Config_DoNotMoveOrDeleteThis` används blir filnamnen till `ResultGenerator` klassen enligt nedan:

- participant.txt
- start_<station_id>.txt
- finish_<station_id>.txt

De två sista filerna fås utav registreringsprogrammet och det kan finns en eller flera. Se appendix A för mer information.

### 3.2.4 Output-filer

Om standardinställningar från `Config_DoNotMoveOrDeleteThis` används så sparas resultatfilerna som skapas utav `ResultGenerator` i mappen "./registered_times" och heter "results_<RaceType>.txt" samt "results_HTML_<RaceType>.html". Filerna är sorterade efter totaltid. Se appendix A för mer information.

## 3.3 Systeminteraktion: Scenario

Denna avdelning ger ett exempel på interaktionen mellan registrerings- och resultatsystemen.

**Exempel: Maratonlopp** Inför ett maratonlopp ansvarar organisatörerna för att manuellt skriva in startnummer och namn i en `participants.txt` fil, på korrekt sätt.

Därefter kan loppet starta: personal (vid separata stationer) registrerar tider med registreringsprogrammet som genererar textfiler.

När loppet är slut samlas alla filer in via USB och lämnas till en dator där resultatprogrammet körs. Resultatprogrammet läser in filerna och genererar resultatfilen. Genom att ange en konfigurationsfil som inparameter till sorteringsprogrammet kan man ange filsökvägar och filnamn om så önskas.

# A Filer och Filformat

Varje fil kan innehålla en beskrivande header följd av riktiga, "semikolon + whitespace"-separerade (; ) data. Exempel för varje fil redovisas här. Filerna kan innehålla annan text om innehåll saknas, se sektion A.2 för mer information om avvikelser.

Alla filer kan även konverteras till HTML-format vid behov.

## A.1 Exempel på filer

Filformat, med exempel, ges för varje fil här. *Headers i italic* är endast beskrivningar datan, d.v.s. den raden finns "fysiskt" med i filen.

**participants.txt**

```
StartNr; Namn
1; Anders Asson
2; Bengt Bsson
3; Chris Csson
4; David Dsson
5; Erik Esson
```

---

**preregtimes.txt**

*StartTime*

```
22.19.39
```

Kommentar: Filen innehåller bara en textrad, som används för back-up i fall av systemcrash.

---

**start_<stationdId>.txt**

<stationId> bestäms av användaren.

*StartNbr; StartTime*

```
13; 22.19.39
```

---

**finish_<stationdId>.txt**

<stationId> bestäms av användaren.

*StartNbr; StartTime*

```
3; 01.03.06
```

---

**`results.txt`** (Maratonlopp)

```
StartNr; Namn; TotalTid; StartTider; Måltider
3; Chris Csson; 01.03.06; 12.02.00; 13.05.06
4; David Dsson; 01.09.07; 12.03.00; 13.12.07
2; Bengt Bsson; 01.14.16; 12.01.00; 13.15.16
5; Erik Esson; 01.12.07; 12.04.00; 13.16.07
1; Anders Asson; 01.23.34; 12.00.00; 13.23.34
```

---

## A.2 Specialfall

Exempel på avvikande fildata:

| **Fil** | **Attribut** | **Avvikelse** | **Orsak** |
|---|---|---|---|
| results.txt | Totaltid | - -.- -.- - | Otillräcklig data för tidsberäkning. |
| results.txt | Starttid | Start? | Saknas starttid. |
| results.txt | Sluttid | Slut? | Saknas sluttid. |
| results.txt | Totaltid | Omöjlig totaltid? | Totaltiden är mindre än 15 minuter. |
| results.txt | Starttid | Flera starttider? | Fler än en starttid har lästs in. |
| results.txt | Namn | Namn Saknas | Deltagare finns inte i participants.txt |
| results.txt | Varvtid | Omöjlig varvtid? | ? |

Table 1: Avvikande data för textfilerna.