פרויקט לקורס תכנות ברשת עם Java – קורס 10332, תשע"ז, סמסטר 1, ד"ר איציק אביב

לסטודנטים ניתנות מחלקות ב-Java עבור מרוץ מכוניות שתוצג בכיתה. החלקות כבר כוללות javaFx ו-mvc.

לקוח:

1. יש לכתוב את המערכת כשרת לקוח
2. כל חלון מרוץ הוא לקוח והשרת מנהל את הלקוחות.
3. כמו כן יש לכתוב את המערכת כמולטי-טרדינג, כל מרוץ הוא טרד.
4. תוצאות המרוץ הן אקראיות (כלומר אין פרמטר חיצוני של מהירות המכונית). מרוץ נמשך כמשך השמעת שיר שמתנגן במהלך המרוץ, השיר מוגרל מתוך מאגר של 3 שירים לפחות.
5. לפני התחלת המרוץ מושמעת הזנקה, וכשמגיעה המכונית האחרונה גם מושמע אות המסמל את סוף המרוץ.
6. כל מרוץ לוקח כמשך זמן השיר ועוד שתי שניות.
7. כל חצי דקה מוגרלים זמני מהירות חדשים עבור כל מכונית במרוץ.
8. כל מרוץ יכיל 5 מכוניות.
9. המכוניות יוצגו בתלת ממד (נושא עצמאי ללמידה).
10. המכוניות יחזרו למסלול כמה פעמים בהתאם לזמן המרוץ.
11. לכל מכונית במרוץ יוגרל סוג (יגואר, מרצדס, פורשה וכולי), צבע, גודל (מיני, רגיל, גדול), צורה (ספורט, סלון, קבריולט), וכאמור מהירות שתשתנה אקראית כל חצי דקה. צורת המכונית בתלת ממד צריכה להתאים פחות או יותר למאפייניה.
12. לכל מרוץ יהיה שם בלעדי שיכיל את מספר המרוץ (מספר רץ מתחילת חיי המערכתי), התאריך והשעה של התחלת המרוץ.
13. לכל מכונית בכול מרוץ יהיה שם בלעדי שיכיל את סוג המכונית וזיהוי המרוץ בו השתתפה.

שרת:

1. לא יכולים להתקיים בעת ובעונה אחת יותר ממרוץ אחד.
2. ייתכן מצב בו לא יתקיים אף מרוץ.
3. מרוץ יכול להתקיים אם מתקיים עבורו שהימרו לפחות על 3 מכוניות מתוך ה-5 המשתתפות במרוץ.
4. כאשר המערכת מזהה שהסתיים מרוץ, או בשלב ההתחלתי שלא התחיל מרוץ, ויש יותר ממרוץ אחד שיכול לצאת לדרך (הימרו על לפחות 3 מכוניות בו), היא תבחר את זה שסכום ההימורים עליו הוא הגבוה יותר.
5. עם עליית המערכת נפתחים 3 מסכים (לקוחות) של מרוצים. אחרי שמרוץ הסתיים, המסך שלו נעלם כעבור דקה (לאפשר למשתתפים לקרוא את התוצאות, שמופיעות גם במסך הסרבר), אם יש מרוץ שעונה על התנאים הוא יוצא לדרך ובנוסף נפתח מרוץ חדש להימורים.
6. המערכת תאפשר, דרך מסך ייעודי, למשתמשים להירשם אליה.
7. כל משתמש יכול להמר על כל מרוץ ובכל מרוץ על כמה מכוניות שהוא רוצה מתוך ה-5, ההימור משמעותו סכום כסף עבור מכונית אחת או יותר במרוץ אחד או יותר.
8. המערכת גובה 5% מכל הימור.
9. מהמר שהמכונית עליה הימר הגיעה למקום הראשון במרוץ מסוים, זוכה בכל הכסף של שאר המהמרים באותו מרוץ (כאמור לאחר ניכוי 5% למערכת מכול הימור). במידה ושניים או יותר מהמהמרים הימרו נכון, הם יתחלקו בכסף לפי יחס ההימור שלהם.
10. כל האירועים וכל המתחרים נרשמים על גבי בסיס נתונים יחסי מסוג MySql שיש לתכנן אותו. היוזר הוא scott והססמא היא tiger.
11. המערכת תתועד לפי כללי התיעוד של javaDoc (נושא ללימוד עצמי).
12. יש לכתוב ממשק שאילתות. המערכת תתמוך לפחות בשאילתות הבאות שתהיינה מוכנות כבר (הוספה תתקבל בברכה):

* היסטוריה:

שליפת המרוצים: כל הפרטים הקשורים במרוצים, המכוניות, התוצאות, המהמרים עליו.

המרוצים יוכלו להיות ממוינים לפי שמם, או תאריכם וזמנם, או לפי סכום ההימורים הכולל למרוץ בסדר יורד.

שליפת מהמרים: כל הפרטים על המהמרים, שמם, היסטוריית ההימורים שלו, כל מרוץ בו הימר, על אלו מכוניות, באיזה סכום, רווחיו באותו מרוץ.

* מצב נוכחי:

בכול רגע ורגע יהיה דיווח מלא לגבי כול מרוץ, למה הוא מחכה כדי לצאת למרוץ, מי המהמרים עליו, על אלו מכוניות הימרו וכמה כל הימור. אותו דבר בסיום כל מרוץ.

* סטטיסטיקות:

סה"כ רווחי המערכת מהמרוצים, ממוין לפי מרוצים בסדר רווחים יורד (עבור כל מרוץ כל פרטיו).

סה"כ רווחי מהמרים (כולל בסימן שלילי), בסדר יורד של רווחי המהמר (עבור כל מהמר כל פרטיו ורווחיו מכול מרוץ מתוך הסך הכול)

דגשים כללים:

יש להשתמש ב-table Views ככול שניתן עבור הצגת הדו"חות.

המערכת תפותח ותוצג ותוגש ב-java8, eclipse mars, javaFx, msql. לא תתקבלנה עבודות עם סביבות פיתוח אחרות.

העבודה תוגש כ-package ששמו יהיה שמות הסטודנטים.

מועד ההגשה – 30 יום לכל המאוחר מסיום הקורס. יש להתקין את המערכת במחשב שלי. קודם להגשה יש לבדוק פורטבליות – מעבר המערכת ממחשב למחשב בלי צורך בשינויי קוד.

הפרויקט יבוצע על ידי צוותים של שני סטודנטים (או בודדים) , הציוות יהיה בתחילת הקורס ויכלול גם עבודה משותפת על התרגילים.

המחלקה הראשית (אותה מריצים) תיקרא כמו ה- package.

Modules of server:

1. DB

* Init DB on server start (if needed)
* Add/update/get data from DB – DBApi

1. Server Logger – client connect/disconnect, race completed
2. Server GUI – Log viewer, Statistics viewer
3. Server API – all services that server can supply to clients
4. Server controller – server behavior, broadcast race
5. Server protocol – accept and handle connections from clients

Modules of client:

1. Client protocol – client connect/disconnect, race completed
2. Client GUI – Login, make bet, view current races status, view running race
3. Client API
4. Client controller

Server open:

Welcome message in logger

Generate list of 3 races

When race become completed, keep race for 2 more minutes, and then remove it from a list and generate a new race instead and send to all clients

When server connects (login page) open active connection

When client disconnects (closes client GUI) close connection

The user cannot open more than one window in parallel

Login request from client:

* If user exists (by id) , get user and send as response
* If user does not exist, insert user and send as response

**DB viewer tab**:

Keep last query

Create enum of supported reports

On “show” button click fetch from DB relevant info

Client open > create connection to server

Client login:

* if user exists, get user getails , open ZRace client GUI and display “welcome back”
* if user does not exist, create user, open Zrace client and display “welcome”
* Before opening ZRace client GUI ask from the server for open races (running or waiting state) – always should get 3 races

Client Zrace:

* On ZRace client GUI display all 3 races + status of each of them
* For each race user is able to view the race (even if not running)
* Bet for race in available only 1 minute before the race start
* The user cannot open more than one window in parallel

Bet:

* Before open bet page, ask from server all user’s bets for the relevant race
* User should not be able to make a bet for the same car in same race several times
* User can put amount of money for several cars in same race ( >0 ) and send a bet to server
* Add input validation for amount fields
* If no amount is specified for car’s field, no bet is done
* If all amount fields are empty, “make a bet” button does nothing (disable button)

Close Client Zrace > send disconnection message to server

General socket: port 9000

When client comes, server gets from imputsteam