תכנות מתקדם 2: 89-211 – מועד ב' תשע"ו

זמן המבחן: **שעתיים**, יש לענות על 6 מתוך 6 שאלות, **בגוף השאלון בלבד**. חומר **סגור**.

בקיאות

שאלה1: (20 נק') לאפליקציית ה web שלנו יש בעיית סקלביליות.

- אהרון טוען שיש לבצע scale out, ואז load balancing באמצעות אלג' round robin
  - כלומר הטלת המשימות (טיפול בבקשות הלקוחות) על השרתים תתבצע ע"פ סבב קבוע.
- ברכה טוענת שיש לבצע scale up, ומכיוון שמדובר ב sticky session עם הלקוח אז ממילא חלוקת המשימות צריכה להתבצע באמצעות session affinity.
  - כלומר אותו השרת שטיפל בלקוח מסוים ימשיך לטפל גם בבקשות הבאות שלו, ולכן אין צורך ב scale out.
- גדעון לא מסכים עם ברכה. הוא לא מאמין ב session affinity והוא טוען שיש צורך ב scale out.
  - כל שרת יוכל לטפל בכל משימה שתהיה, כי את ה data של ה session נשמור בשרת אחר המהווה central session store, וכך נוכל לבצע load balancing טוב יותר!
- דקלה מציעה גם scale out וגם session affinity באופן הבא:
  - כבר בבקשה הראשונה נשמור ב cookie אצל הלקוח את ה ID של ה session. בכל בקשה נוספת מהלקוח הוא יעביר את המידע שב cookie וכך נוכל להקצות את המשימה אל השרת שטיפל בו בעבר.
- הרשלה טוען שיש לבצע scale out, ו IP Address Affinity.
  - בהינתן בקשת לקוח, נריץ hash על ה IP שלו, והתוצאה מודולו N תקבע מי מתוך N השרתים צריך לטפל בבקשה שלו.

עבור על אחד תארו בקצרה מהם היתרונות והחסרונות בשיטה שהציע (5 נק' לכל שיטה).

אהרון יתרונות:_____

_____

חסרונות: _____

_____

ברכה יתרונות:_____

_____

חסרונות: _____

_____

גדעון יתרונות:_____

_____

חסרונות: _____

_____

דקלה יתרונות:_____

_____

חסרונות: _____

_____

הרשלה יתרונות:_____

_____

חסרונות: _____

_____

שאלה 2: (12 נק')

הקיפו בעיגול את התשובות הנכונות:

א. ניתן באמצעות double check locking להתגבר על בעיית ה singleton בסביבה שהיא multithreaded.
ב. MVVM ניתן ממש רק ב NET. בטכנולוגיית WPF.
ג. Service Locator מהווה Runtime linker
ד. ב MVP שכבות המודל וה View צריכות להכיר רק את עצמן.

מיומנות עיצוב קוד (Design) וכתיבת קוד

שאלה 3: (20 נק')

לצורך איתור ואבחון תקלות ברכבים אנו ממדלים את ההתנהגות הצפויה של כל רכיב ורכיב – כל רכיב (Component) זוכה לאובייקט משלו. המחלקה WarningLight תומכת בפעולות ()on, ()off ו ()blink. יש לה מימושים שונים כמו LowFuel, CheckEngine וכו'. כאשר רכיב כלשהו הופך לתקול אז נורית האזהרה צריכה לפעול בהתאם. לדוגמא, כאשר הרכיב TimingChain הופך לתקול אז ה CheckEngine צריכה להיות דולקת, וה OverdriveOff צריכה להבהב. באופן דומה גם אם המנוע תקול אז CheckEngine צריכה לדלוק. לכל נורת אזהרה קיימים תנאים שונים שתחתם עליה לפעול בצורה הרצויה.

המתכנתים של המחלקות מסוג WarningLight יודעים להגדיר את התנאים לפעולה הרצויה (להידלק, להבהב, או להיכבות) אך הם אינם יודעים *מתי* ליזום את הפעולה הרצויה (כלומר מתי הרכיב הופך לתקול)

עליכם **לשרטט** תרשים מחלקות (class diagram) **ב UML**, המציג עיצוב שמאפשר ליזום את הפעולה הרצויה של כל נורית אזהרה כאשר הרכיבים הרלוונטיים הופכים לתקולים (ללא busy waiting).

תשובה:

שאלה 4: ממשו בקוד (באיזו שפה מונחית עצמים שתרצו) את הפתרון העיצובי של שאלה 3. (16 נק')

תשובה:

שאלה 5: (16 נקודות)

נותנה המחלקה PiCalculator שממשה את המתודה double calcPi(int digits) המחשבת את פאי עד ל digits ספרות לאחר הנקודה. לצערנו זה עלול לקחת לא מעט זמן וזה קוד סגור שלא ניתן לשינוים. עליכם **לשרטט** תרשים מחלקות (class diagram) **ב UML**, המציג עיצוב שמאפשר

- להריץ את calcPi בצורה אסינכרונית
- ולטפל בערך המוחזר כך ש
  - הלקוח יוכל לשלוף אותו בעת הצורך
  - ורק אם הלקוח מנסה לשלוף את הערך לפני הזמן הוא יאלץ להמתין עד לסוף החישוב.

תשובה:

שאלה 6: ממשו בקוד (באיזו שפה מונחית עצמים שתרצו) את העיצוב של שאלה 5 (16 נק')

בהצלחה!

תשובה 6: