# **תרגיל 2 – משחק צוללות ב JavaFX**

## מגישים:

1. אור מנצור – 204311997 (or.mantzur@gmail.com)
2. דודי יחזקאל – 200441749 (dudi.2705@gmail.com)

## הערות לבדיקת התרגיל:

קבצי מקור:  
כל קבצי קוד המקור ממוקמים תחת תיקיית Source Code.

מיקום קובץ Bat:  
קובץ הBat ממוקם בתיקיה יחד עם הקובץ jar של הלוגיקה וזאת בכדי שקבצי שמירת המשחק ימוקמו   
באותה התיקייה.

בדיקות המבוצעות על קובץ הxml :

1. בהרצת פקודת טעינת המשחק (פקודה 1):
   1. טעינת קובץ
      1. הקובץ שניתן קיים
      2. הקובץ שניתן אינו תיקייה
      3. הקובץ הוא מסוג xml
   2. בדיקת תקלות JAXB
   3. בדיקה שגודל הלוח הוא בין 5-20
   4. בדיקת **סוגי** הספינות שיש למקם על הלוח
      1. בדיקה שרשימת סוגי הספינות אינה ריקה
      2. אורך הספינה הוא מספר אי שלילי
      3. הscore של הספינה הוא מספר אי שלילי
      4. אין שתי ספינות עם אותו ה ID
2. בדיקות לוגיות המבוצעות על קובץ הxml בהרצת פקודת תחילת משחק:
   1. בדיקה שאין שחקן שהוא null (ביצירת משחק חדש שולחים את שני השחקנים בctor)
   2. בדיקה שאכן קיים קובץ שנטען מxml (חוסם את האפשרות להתחיל מחדש משחק שנטען משמירה)
   3. בדיקה לוגית שאין התנגשויות בהוספת הספינות ללוח (אין התנגשות של ספינות, אין ספינות סמוכות, אין ספינות שממוקמות מחוץ ללוח)
   4. כמות הספינות המוגדרת לכל לוח תואמת את כמות הספינות שמוקמו בכל לוח.

## הסבר כללי על אופן מימוש התרגיל:

### ישנם שני מודולים בתרגיל זה

1. BattleshipsGameLogic - לוגיקת המשחק.   
   ממומש באופן כזה שניתן יהיה לעשות בו שימוש מכל UI שנבחר, אותו המודול מתרגיל 1 עם שינויים מינוריים.
2. BattleshipsJavaFX - ה-UI של התרגיל עבור משחק על גבי ה JavaFX.   
   מכיל את נקודת הכניסה (Main) של המשחק והוא עושה שימוש במודול של הלוגיקה המתואר לעיל.

### תיאור המחלקות העיקריות בכל מודול:

**BattleshipsGameLogic**

1. GamesManager

מחלקה זו מנהלת את המשחק (בתרגילים הבאים תנהל גם יותר ממשחק אחד).   
מחלקה זו מממשת את הממשק IGamesLogic שתתואר בהמשך.

* 1. Data Members:
     1. כל השחקנים הרשומים למשחק (בתרגיל זה יש רק שניים).
     2. כל המשחקים הקיימים במשחק (בתרגיל זה יש רק אחד).
  2. Methods:
     1. המתודות שיתוארו בממשק IGamesLogic.

1. IGamesLogic:

ממשק אשר מחצין את המתודות הרלוונטיות בלוגיה לUI.

הסיבה העיקרית לקיום מחלקה זו היא בכדי לאפשר עבודה זהה עם הלוגיקה בין כל UI עתידי.

בנוסף, מחלקה זו מפשטת את השימוש במודול זה ע"י חשיפת המתודות הבאות:

* 1. טעינת קובץ XML עבור הגדרות המשחק
  2. התחלת משחק
  3. תקיפת משבצת בלוח השחקן
  4. קבלת זמן המשחק עד כה
  5. הצבת מוקש
  6. יצאה מסודרת מהמשחק

1. Game:
   1. Data Members:
      1. הגדרות המשחק
      2. שני השחקנים
   2. Methods:
      1. initGame – אתחול המשחק הכולל בתוכו בדיקה לוגית של כל הגדרות המשחק.

לדוגמא, בדיקה שאין חפיפה בין הספינות, שכל הספינות שמוגדרות בקובץ אכן נמצאות על כל אחד מלוחות השחקנים וכו'.

* + 1. resetGame – אתחול מחדש עבור משחק קיים, איפוס לוחות השחקנים למצב הראשוני. במידה והשחקן רוצה לסיים את המשחק הנוכחי, אך רוצה להשתמש באותו משחק ששיחק.

1. GameSettings:
   1. Data Members:
      1. גודל הלוח
      2. מספר מוקשים לשחקן
      3. סוג המשחק
      4. סוג הספינות שיש לטעון ומספרם
   2. Methods:
      1. loadGameFile – טעינת קובץ XML עבור הגדרות המשחק מהמשתמש ובדיקה ראשונית של תקינות ההגדרות. לדוגמא, בדיקה שגודל הלוח הוא מספר תקין בטווח, בדיקה שהלוח לא ריק מספינות, בדיקה שאין שני סוגי ספינות שונות בעלות או מזהה יחודי וכו'.
2. Player:
   1. Data Members:
      1. ID
      2. שם
      3. לוח המשחק
      4. לוח המשחק של היריב
      5. נקודות במשחק
      6. זמן משחק
      7. מספר התורות ששוחקו
   2. Methods:
      1. attack – תקיפה של משבצת בלוח. מתודה זו מחזירה את תוצאת התקיפה. לדוגמא: השחקן פגע בצוללת, מוקש, משבצת שכבר תקף בעבר וכו'.
      2. plantMine – הטמנת מוקש בלוח ספינות השחקן (בונוס – יפורט בהמשך).
3. Board:
   1. Data Members:
      1. לוח המשחק
      2. רשימת ספינות
   2. Methods:
      1. addShipToBoard – הוספת ספינות ללוח המשחק בעת אתחול הלוח.
      2. allShipsWereSunk – בכל שלב במשחק ניתן לבדוק האם כל הספינות שעל הלוח הוטבעו.
4. BoardCell:

תא בלוח המשחק. Board הוא למעשה מערך דו מימדי של BoardCells

* 1. Data Members:
     1. מיקום התא
     2. האובייקט שנמצא בתא (ספינה, מוקש, מים)

1. GameObject:

מחלקה אבסטרקטית אשר ממנה יורשים כל האובייקטים על הלוח (ספינה, מוקש, מים).

* 1. Data Members:
     1. מיקום על הלוח
  2. Methods:
     1. getAttackResult – מתודה אבסטרקטית המבצעת את הפעולות הנדרשות בפגיעה באובייקט (בדיקה האם הספינה הוטבעה לדוגמא) ומחזירה את תוצאת הפגיעה

**BattleshipsJavaFX**

**הסבר כללי:**

מודול זה בנוי באופן הבא:

1. JavaFXManager – המחלקה המרכזית במודול אשר משמשת כ"מתווך" בין החלק הלוגי ל UI
2. view (package) – מכיל את כל החלונות ואת הcontrols שלהם
3. model (package) – מחלקות לוגיות נוספות שמימשנו לטובת הUI אך אינן חלק מהמימוש הלוגי של המשחק לדוגמא:
   1. ImageViewProxy המשמש כproxy לעבודה עם ImageView
   2. ניהול הלוגיקה של ביצוע replay בסיום משחק.
4. styles (package) – כל קבצי הcss השונים של המשחק

**תיאור המחלקות העיקריות:**

1. JavaFXManager:

מחלקה אשר מנהלת את ה UI של המשחק באמצעות JavaFX.

* 1. DataMembers:
     1. MainWindow
     2. PauseWindow
     3. PlayerInitializerController
     4. רשימה של כל המהלכים שבוצעו(לטובת האופציה של שחזור משחק)
  2. Methods:
     1. הצגת MainWindow
     2. הצגת PauseWindow
     3. טעינת משחק
     4. התחלת משחק
     5. ביצוע מהלך
     6. שחזור משחק
     7. סיום משחק

1. MainWindowController:  
   החלון הראשי של המשחק, אשר מכיל את הלוחות, הסטטיסטיקה של המשחק, ביצוע מהלכים ועוד.
2. PauseWindowController:  
   חלון זה מוצג בשלושה מצבים:
   1. ברגע שפתחתנו את האפליקציה
   2. כאשר עצרנו את המשחק הנוכחי
   3. בסיום משחק

חלון זה מאפשר:

1. לטעון משחק חדש
2. להתחיל את המשחק הנוכחי שטעון כבר מחדש
3. לחזור למשחק(במידה ועצרנו את המשחק באמצע)
4. סיום המצב הנוכחי
5. שחזור המשחק(אופציה זו קיימת אם עצרנו את המשחק באמצע וגם בסיום המשחק)
6. יציאה מהאפליקציה
7. PlayerInitializerController:

בעת אתחול המשחק, ניתן למשתמש אופציה לבחור את שמו במשחק ולהוסיף תמונה.

1. BoardAdapter:

תפקידה של מחלקה זו היא ליצור את הלוחות של השחקן ולעדכן את התמונות של כל תא בהתאם.

1. ImageViewProxy:

מחזיקה את התא הלוגי של הלוח ומעדכנת את התצוגה שלו בהתאם.

1. ReplayGame:

כל אובייקט של המחלקה מייצג מהלך אחד במשחק. אובייקט זה שומר את התא שהותקף באותו מהלך ואת תוצאת הפגיעה.

## **פירוט בונוסים שמומשו**