מכללת הדסה, החוג למדעי המחשב
מבוא לתכנות מונחה עצמים והנדסת תוכנה
סמסטר א', תשפ"ד

פרויקט סוף סמסטר

תאריכי הגשה: ראו לוח זמנים בהמשך.

מטרת התרגיל:

בתרגיל זה נממש שוב את המשחק "חתול ועכבר" שמימשנו בתרגיל 2. ההבדלים המרכזיים הם שהפעם נשתמש בירושה ופולימורפיזם וכן נשתמש בספרייה הגרפית SFML.

מאז תרגיל 2 למדנו נושאים חדשים והתקדמנו באופן כללי בתכנות מונחה עצמים, כעת נוכל לעצב תיכון טוב יותר, יפה יותר ויעיל יותר של האפליקציה בהתאמה לעקרונות תכנות מונחה עצמים.

כמו כן, נוסיף תכונות נוספות לעומת מה שעשינו בתרגיל 2.

שינויים והוספות:

רוב פרטי המשחק נשארים בדיוק כפי שהיו בתרגיל 2, למעט מה שכבר הוזכר לעיל, שימוש בירושה, פולימורפיזם וממשק גרפי, וכן השינויים וההוספות המפורטים להלן.

משחק מונחה שעון:
המשחק כעת יהיה "בזמן אמת", מונחה שעון, ולא מונחה תורים. ראו פירוט בהמשך, בפרט לגבי מהירות התזוזה.

התנגשות באריחים או בין דמויות:
הגדרת התנגשות היא כאשר יש נקודה חופפת בין המלבנים החוסמים הרלוונטיים, כלומר המלבן החוסם את האריח והמלבן החוסם את הדמות או המלבנים החוסמים של שתי הדמויות המתנגשות.

הגבלת זמן:

נוסיף כעת בחלק מהשלבים הגבלת זמן. הבחירה באלו שלבים תהיה הגבלת זמן יכולה להיות רנדומלית או להיקבע מראש בקובץ כחלק מהמידע לגבי השלב, וכן כמות הזמן המוקצבת יכולה להיות רנדומלית (אבל חשוב לוודא שמדובר בכמות סבירה שתאפשר את סיום השלב אבל גם תכניס עניין למשחק, כך שזה לא לגמרי רנדומלי) או קבועה מראש בקובץ. בשורת המידע יופיע כעת גם טיימר שיספור לאחור, כפי שמוזכר להלן. כשהטיימר מתאפס – השחקן נפסל וצריך להתחיל את השלב מחדש (מאופס לחלוטין, כפי שהשלב מוגדר בקובץ, כלומר דלתות שנפתחו תהיינה סגורות, גבינה שנאכלה חוזרת וכו'), בשונה מפסילה שקורית בגלל התנגשות בחתול, שאז רק מיקומי הדמויות מתאפסים אבל מצב שאר האובייקטים נשאר כפי שהוא.

"מתנות":

בנוסף למתנה שכבר הוגדרה בתרגיל 2, שמעלימה את אחד החתולים, נוסיף סוגי "מתנות" נוספים. סוגי המתנות הנדרשים הם:

1. הגדלת הזמן המוקצב (בהנחה שיש בשלב הזה מגבלה על הזמן)
2. הקפאה זמנית של החתולים (להחלטתכם מה קורה בהתנגשות בהם במצב הזה)
3. הוספת חיים

תוכלו להחליט האם התוספות קבועות מראש או רנדומליות. ההתנהגות לגבי התנגשויות תהיה בכל המתנות כמו במתנה שהוגדרה בתרגיל 2, החתולים יכולים לעבור במקום שיש בו מתנה, אבל הם כמובן לא יכולים לקחת אותה.

בחרו מגבלת זמן הגיונית להקפאת החתולים.

שלבי המשחק:

עליכם לספק לפחות שלושה שלבים (קל יותר עכשיו שיש לנו עורך גרפי :)).

כמו בתרגיל 2, גם פה נדרש שתהיה דרך להוסיף שלב חדש בלי צורך להוסיף את שם הקובץ החדש לקוד. הדרכים המוצעות הן:

1. החזקת כל השלבים בקובץ אחד רצוף כך שאפשר להוסיף לסופו שלבים נוספים.
2. שימוש בתבנית מסוימת לשמות הקבצים, כך שהתוכנית יודעת אוטומטית מה שם הקובץ הבא שהיא צריכה לנסות.
3. שימוש בקובץ שממנו קוראים את רשימת שמות קובצי השלבים ("playlist").

באופן כללי, ניתן להניח שאם הקובץ קיים, הוא תקין.

שימוש בתפריט:

המשחק צריך לכלול תפריט בסיסי עם אפשרויות של בחירת משחק חדש, הצגת עזרה (שתציג את הוראות המשחק) או יציאה. התפריט יוצג בתחילת ריצת התכנית או אחרי שהמשחק נגמר. לבחירתכם האם התפריט יופעל בעזרת העכבר או המקלדת (או שניהם). אם הוא מופעל בעזרת המקלדת ראוי לתמוך בבחירת אפשרות מהתפריט על ידי לחיצה על אות מתוך הכיתוב המופיע על הכפתור. במקרה כזה, נדגיש את האות על ידי קו תחתי (כמו בדוגמה כאן: File Edit View Window Help).

תצוגת מידע:

בשורת המידע נציג לשחקן את מספר השלב, ואת הזמן שעבר מתחילת השלב. אם יש מגבלת זמן בשלב הנוכחי, יופיע טיימר שסופר לאחור במקום זה שסופר קדימה או בנוסף לו. בנוסף, יופיע מספר הנקודות שהוא צבר, כמה חיים נותרו לו ומספר המפתחות שברשותו.

תזוזה:

למרות שבשימוש במסך גרפי ניתן להזיז את הדמויות גם באלכסון, לכאורה, בכל זאת נשאיר את התזוזות רק בקווים ישרים לאורך הצירים, ולכן גם המקשים של השחקן נשארים כפי שהם (חיצים לתזוזה לאחד מארבעת הצדדים בהתאמה; אין משמעות למקש הרווח כי המשחק לא מתנהל לפי תור).

הלחיצה על המקשים נשארת כשהייתה (רק שהפעם זיהוי הלחיצה קורה בעזרת אירועים של SFML במקום בעזרת conio). ההבדל הוא שעכשיו המשחק מתנהל לפי השעון. ממילא, צריך לשים לב שקצב התזוזה לא יסתמך על קצב קבלת האירועים של key pressed, שאיננו אחיד, אלא על משך הלחיצה ועל השעון. הדרך לעשות זאת היא כך שהלחיצה רק קובעת האם הדמות הנוכחית תזוז ולאיזה כיוון היא תזוז כאשר נרצה להזיז אותה, אבל התזוזה בפועל מתבצעת לפי הזמן שחולף (בדומה לדוגמה במצגת מהתרגול). כדי לעצור את הדמות (גם בלי להיתקע במכשול), האפשרות הפשוטה היא לעצור ברגע שהשחקן הפסיק ללחוץ על המקש (אירוע של key released).

מהירויות תזוזה:

עליכם להחליט כיצד לעדכן את מיקום הדמויות. מכיוון שהגדרנו שהמשחק מונחה שעון, הרי שתצטרכו לבצע זאת כל פרק זמן מסוים ותצטרכו לקבוע בכמה הדמות זזה בכל פרק זמן כזה, לפי הכיוון שאליה היא זזה כרגע. האם עדכון כל עשירית שנייה יהיה מוצלח? אולי פחות? אולי יותר? האם תזוזה של פיקסל או שניים סבירה? האם, כדי לתת די אתגר, האויבים צריכים להיות מהירים יותר (כלומר, יותר פיקסלים בכל איטרציה) מהשחקן? עד כמה מהירים? כל כמה זמן האויבים יחשבו מחדש את הכיוון שלהם? לא נקבע כללים בנושא, אלא נשאיר זאת עבורכם לנסות ערכים שונים ולהחליט איך ליצור משחקיות טובה. בכל מקרה, תעדו את החלטותיכם בקובץ ה־Readme.

עיצוב האובייקטים:

לגרפיקה מוצלחת כנראה שתרצו לצייר את האובייקטים בתוכנה חיצונית או למצוא תמונות מתאימות ולטעון את התמונות בעזרת sf::Texture. במצב כזה, כדאי שהתמונה תהיה עם רקע שקוף (ניתן לעשות זאת בתמונות ששמורות כ־png, למשל), כדי למנוע רקע לבן שיתווסף לדמות. זה חשוב בעיקר אם מדובר בדמות שאין צורה פשוטה שתכיל אותה וצריך להכיל אותה בתוך sf::RectangleShape או sf::Sprite שחוסם אותה. כמו כן, כדאי לדאוג שהדמות תתפוס ככל האפשר כמה שיותר מהמלבן החוסם, כדי שיהיה ברור יותר בשעת המשחק מתי נחשב שדמויות התנגשו זו בזו או במכשול.

סאונד:

אתם נדרשים להוסיף למשחק סאונד.

הערות על גודל החלון וה"משבצות":

- חלקכם אולי תגלו את האפשרות להציג את החלון על מסך מלא (מצב fullscreen). שימו לב שהמצב הזה בעייתי מאוד מבחינת הצגת הפרויקט, בפרט אם נצטרך מסיבה כלשהי לבצע אותה דרך זום. גם אם בחרתם לתמוך במצב הזה בפרויקט שלכם, וודאו שניתן לשנות זאת בקלות (בלי שינויים בקוד!) כך שאפשר יהיה להריץ את המשחק גם בחלון רגיל.
- הקוד צריך לתמוך בגדלי שלבים שונים. איך נתאים את גודל השלב לגודל החלון?
  - יש ל-SFML כלי שיכול לעזור פה (sf::View / viewport), אבל לא נספיק ללמוד עליו הסמסטר. כמו תמיד, קיימת האפשרות ללימוד עצמאי ושימוש בכלי הזה, וכרגיל, השימוש צריך להיעשות בצורה הנכונה.
  - כרגע, האפשרות הכי פשוטה למימוש היא להשתמש במלבנים בגודל קבוע ולהתאים את גודל החלון לגודל הרצוי לפי גודל השלב. שימו לב לא להשתמש ב-setSize של החלון כדי לשנות את הגודל שלו (כי נקבל התנהגות שנראית מוזרה, כל עוד לא למדנו את הכלי הנ"ל שלא נספיק לדבר עליו), אלא ב-create (שמקבלת פרמטרים כמו הבנאי וגורמת למעשה להריסה של החלון הקודם ויצירה של חלון חדש).
  - כמובן שאם תרצו לנהל חישובים ולהתאים את גודל הטקסטורות לגודל מלבנים משתנה (בעיקר בעזרת scale, מן הסתם), ולהשאיר את החלון בגודל קבוע במשך כל ריצת המשחק, זה יהיה פתרון יפה יותר.
- לפעמים נגלה שלשחקן קשה להיכנס בין שני קירות וכדומה. הסיבה היא כי אם הוא לא בדיוק מתאים לרווח הזה, מספיקה תזוזה של פיקסל לכאן או לכאן כדי לגרום לו להתנגש בקיר כדי שהוא לא ייכנס. קל לזהות שזה המצב אם נוסיף קצת הדפסות או ניעזר בדיבאגר כדי לראות מה המיקום שלו ומול מה הוא מתנגש. קיימים מספר פתרונות אפשריים לבעיה הזו.
  - הפתרון הכי פשטני הוא להגדיר את השחקן (ויזואלית או רק לצורך חישוב ההתנגשויות) כך שהוא קצת יותר קטן מגודל משבצת במשחק, כך שהמסדרונות האלה הופכים להיות מספיק רחבים בשבילו.
  - פתרון יפה הוא לזהות את המצב שיש הבדל של פיקסל או שניים ו"ליישר אותו" מעלה/מטה/ימינה/שמאלה (כתלות בכיוון ההתנגשות) כך שהוא יתאים ל"משבצת".

הערות:

- בניגוד לתרגיל 3, הפעם אין איסור להחזיק במחלקות משתנים מסוג הצורות של SFML. להיפך, עדיף להחזיק את הצורות מוכנות כבר, וכמו כן לחשוב אלו מאפיינים (למשל, מיקום) כבר אין צורך להחזיק בנפרד כי הם חלק ממאפייני הצורה.

- למעקב אחרי הזמן החולף – המחלקה הרלוונטית היא sf::Clock. אפשרות אחרת היא להשתמש בספריית std::chrono שמשולבת כחלק מהספרייה הסטנדרטית מאז C++11. שימו לב שאופן השימוש הוא על ידי בדיקה כמה זמן עבר (בעזרת הפונקציה getElapsedTime()), במקרה של sf::Clock, או באמצעים מתאימים במקרה של שימוש בספרייה הסטנדרטית) ושימוש במידע הזה לקבלת החלטות או לחישוב הזזות וכדומה. מן הסתם המקום המתאים לחישובים האלה הוא בלולאת האירועים או בסמוך לה. (ואגב לולאת האירועים, נזכיר שכדי להתנהל לפי השעון יש צורך להשתמש הפעם ב־pollEvent() ולא ב־waitEvent()).
- הקפידו על תיכון נכון, מונחה עצמים, תוך שימוש ראוי בירושה **ובפולימורפיזם**. שימו לב, לא מספיק רק להשתמש בירושה. אתם נדרשים להשתמש בפולימורפיזם כראוי, ולהקפיד על חלוקת אחריות נכונה בין המחלקות.
- כרגיל, יש לתעד בקובץ ה־Readme תוספות מיוחדות או החלטות לגבי דברים שלא מוגדרים בפירוש.
- כפי שהוזכר כבר בתרגיל 2, ונכון עוד יותר עבור הפרויקט, בכל פרויקט מספיק מורכב יש התנהגויות שלא מוגדרות במדויק על ידי הדרישות. אפשר להתייעץ במקרה הצורך, אבל ההנחייה הכללית היא שעל דברים שלא נאמרו בהגדרות אתם מוזמנים להחליט בעצמכם מה הדרך הטובה יותר לביצוע הפעולה או לפתרון הבעיה (כלומר, טובה יותר מבחינת משחקיות, או סבירה מבחינת משחקיות וקלה מבחינה תכנותית :)).
- אתם רשאים כמובן להוסיף על הרמה הבסיסית, הן מבחינה גרפית, הן תוספות לשיפור המשחקיות (סוגים נוספים של "מתנות", אויבים חכמים יותר וכדו') וכן בשימוש באלגוריתמים מתוחכמים או בתבניות עיצוב וכלי תכנות מתקדמים. בכל מקרה, אם הוספתם – כדאי לציין זאת בקובץ ה־Readme.
- אם השתמשתם בקוד פתוח מהאינטרנט חובה עליכם לציין זאת בקובץ ה־Readme בתוספת קישור למקור. זיהוי של קטעי קוד זהים אצל צוותים שונים ללא ציון המקור, עלול להוביל לתלונה משמעתית. בכל מקרה, יידרש מכם להפגין הבנה במטרת הקוד הזה והשימוש הנכון בו (גם אם לא הבנה מדויקת של המימוש, הרי אחרת הייתם כותבים אותו בעצמכם מלכתחילה).

הגשת תיכון:

הגשת התיכון נעשית בהגשת דיאגרמת UML (מסוג class diagram) המציגה את המחלקות המתוכננות עם מאפיינים (שדות) עיקריים ופונקציות עיקריות שלהן ועם הקשרים בין המחלקות (לפחות ירושה, אבל עדיף גם הכלה). בנוסף, יוגש הסבר מילולי קצר שיסייע בהבנת הדיאגרמה וחלוקת האחריות בין המחלקות. ההסבר המילולי יכלול הסבר קצר על כל מחלקה ואם יש צורך – גם על הקשר בין מחלקות שונות.

הקפידו על שמות משמעותיים גם למחלקות, גם לפונקציות המחלקה וגם למשתנים, כך שיהיה ברור מקריאת התיכון מה התפקיד של כל דבר.

חשוב להבין: בלוח הזמנים מוקדש זמן יחסית קצר עבור הגשת התיכון, מכיוון שמטרתו העיקרית היא לגרום לכם לתכנן לפני שתיגשו לממש את הפרויקט. ההערכה היא שלא תידרשנה לשם כך יותר ממספר שעות במקרה

הגרוע ביותר. אין צורך לתכנן כאן את הקוד לפרטי פרטים, אלא רק לשבת ולחשוב איך נראה כרגע שכדאי יהיה לממש את הדברים מבחינת חלוקת פונקציונליות בין המחלקות, מבחינת עצי ירושה וכדומה.

התיכון צריך להיות סביר והגיוני, אבל לא יקרה שום אסון אם במהלך כתיבת הפרויקט בפועל תחליטו לשנות את חלוקת המחלקות שלכם וכדומה. זה לא אומר שזה בסדר אם התיכון יהיה מרושל מלכתחילה; זה כן אומר שלא צריך להילחץ אתו יותר מידי.

קיימות מספר שיטות מומלצות כדי לשרטט את הדיאגרמה. הסברים מפורטים נמצא באתר בקובץ ההנחיות ליצירת Class Diagram (UML).

ההסבר המילולי יהיה בקובץ Word. זכרו, עדיף קובץ בעברית ברורה על פני קובץ באנגלית לא ברורה!

הגשת התיכון תתבצע כרגיל, כשהקבצים שאתם מגישים מכווצים לקובץ zip, ושם הקובץ, כרגיל, כולל את שמות המגישים אלא שהפעם במקום שם התרגיל תופיע המילה design, כלומר תבנית שם הקובץ היא oop1_design-firstname_lastname.zip, עבור מגיש בודד, כאשר firstname_lastname הוא השם המלא (לדוגמא אלברט איינשטיין יגיש את התיכון כך: oop1_design-albert_einstein.zip). במקרה של הגשה בזוג, שם הקובץ יהיה לפי התבנית oop1_design-firstname1_lastname1-firstname2_lastname2.zip, עם שמות המגישים בהתאמה (ללא רווחים; כלומר, גם בשמות עצמם יש להחליף רווחים בקו תחתי ובין שמות חברי הצוות השונים יפריד מקף, כפי המודגם לעיל). כמו כן, במקרה של הגשה בזוג, **רק אחד** מהמגישים יגיש את הקובץ ולא שניהם.

הצגת השלד:

כדי לגרום לכם להתקדם עם הפרויקט ולנצל את הזמן ולא לדחות הכול לסוף, כמו שעלול לקרות לרבים מאתנו, אתם נדרשים להציג התקדמות אחרי כשבועיים מהגשת התיכון. הצגת השלד תיעשה בזמן הסדנה. כל צוות ישב עם המתרגל במשך 5-10 דקות ויציג את ההתקדמות הנוכחית והניקוד לשלב הזה ייקבע בהתאם.

אנחנו מודעים לכך שזה לא הרבה זמן אחרי תחילת הכתיבה, אבל כן נצפה לראות שלד. העדיפות היא אם ניתן להציג חלק מסוים שעובד כבר, אבל בכל מקרה נצפה לראות התקדמות מסוג כלשהו ולא רק את ההצהרות על המחלקות שבתכנון.

מה נצפה לראות בהצגת השלד?

- משחקיות בסיסית פעילה: הקוד מתקמפל ורץ, נפתח חלון, מוצגים לפחות חלק מהאובייקטים מהלוח, יש לפחות דמות אחת (מתוך הדמויות הנעות, כלומר הדמויות של השחקן או "של המחשב") ויש לה תנועה בסיסית (שחקן – על ידי המקלדת, דמות "של המחשב" – תזוזה עצמאית), אבל לא בהכרח שיש ניהול התנגשויות (למשל, ייתכן שהדמויות עוברות דרך הקירות).

- תיכון ראוי: חלוקה סבירה למחלקות, שימוש נכון בירושה ובהכלה, חלוקת אחריות בין המחלקות (לא שהקונטרולר עושה הכול ובלי שום לוגיקה במחלקות של האובייקטים האחרים במשחק), אנקפסולציה ראויה.
- תכנות נאות: שמות משמעותיים, פונקציות באורך סביר, קוד בהיר.

הצגת הפרויקט:

בזמן שייקבע, כל צוות בתורו יציג את הפרויקט בפני סגל הקורס (המרצה והמתרגל/ת הרלוונטיים). בשלב זה הפרויקט כבר הסתיים והוגש. הודעה סופית על מועד הצגת הפרויקט תבוא בהמשך. אי הגעה להצגת הפרויקט ללא אישור מוקדם (לא באותו היום!) מסגל הקורס, תפחית 5 נקודות מציון הפרויקט הכולל.

הערות:

- צוות של שני סטודנטים יקבל בדרך כלל את אותו הציון לפרויקט חוץ מאשר מקרים חריגים בהם יהיה פער משמעותי בין תרומתם לפרויקט. על כן חשוב ששני הסטודנטים ייטלו חלק פעיל בהצגה, בהבנת הפרטים ובהסברים.
- הצגת הפרויקט הוא כלי משמעותי עבורכם כדי להסביר ולהגן על עבודתכם. צוות שלא יגיע להצגה יפסיד את ההזדמנות החשובה הזו. בנוסף יופחתו 5 נקודות מציון הפרויקט למי שלא יגיע להצגה, כאמור לעיל.
- לא ניתן לקבל דחיה להצגה או להגשת הפרויקט, וימי האיחור והמתנה שהיו נהוגים בתרגילים אינם תקפים לפרויקט. מקרים חריגים באופן מיוחד יפנו למרצה.
- הערכת הפרויקט הינה הערכה ולא שקלול של נקודות. ההערכה מבוססת על ארבעה אנשי צוות: מרצה, מתרגל ושני בודקי תרגילים ומתייחסת למדדים הבאים: רמת המשחקיות, היקף ועומק, רמת התכנות מונחה העצמים, רמת התכנות הכללית, רמת ההבנה והבקיאות בפרטים.
- לא ניתן לערער על ציון הפרויקט חוץ מאשר במקרים חריגים של טעות בהערות או בשקלול. כמו תמיד, בדיקה חוזרת עשויה להעלות או להפחית את הציון.
- הערכה של פרויקט שעונה על מרבית הדרישות הוא 85. תוספות מהותיות ורמת תכנות גבוהה באופן משמעותי תוסיף לציון. חוסר שלמות ובעיות אחרות יפחיתו מהציון.

הגשת הפרויקט:

את הפרויקט מגישים באופן דומה לשאר התרגילים (רק עם השם project במקום exN, כמפורט למטה). כמו תמיד, הנחיות הגשה מפורטות נמצאות בסוף הקובץ הזה.

לוח זמנים:

(בסוגריים מצוין פרק הזמן מאז נקודת הציון הקודמת.)

קמפוס הנביאים:

- תחילת הפרויקט – יום ראשון, 11/02/2024 (יום הלימודים הראשון אחרי הגשת תרגיל 3)
- הגשת תיכון (תוך שלושה ימים) – יום שלישי, 13/02/2024 (עד השעה 23:59)

- הצגת השלד (אחרי שבוע) – ימים שלישי–רביעי, 20-21/02/2024, בזמן התרגול והסדנה
- הגשת הפרויקט (אחרי שבוע) – יום חמישי, 29/02/2024 (עד שעה 23:59) – לא סופי!
- הצגת הפרויקט (בשבוע שאחרי ההגשה) – ימים ראשון–שלישי 03-05/03/2024 – לא סופי!

קמפוס שטראוס גברים:

- תחילת הפרויקט – יום ראשון, 11/02/2024 (מיד לאחר הגשת תרגיל 3)
- הגשת תיכון (תוך שלושה ימים) – יום שלישי, 13/02/2024 (עד שעה 23:59)
- הצגת השלד (אחרי שבוע) – יום שלישי, 20/02/2024, בזמן הסדנה
- הגשת הפרויקט (אחרי שבוע וקצת) – יום חמישי, 29/02/2024 (עד שעה 23:59) - לא סופי!
- הצגת הפרויקט (בשבוע שאחרי ההגשה) – יום ראשון, 03/02/2024 - לא סופי!

ניקוד:

- הגשת התיכון – 5%
- הצגת השלד – 10%
- הצגת הפרויקט והגשתו – 85%

קובץ ה־README

יש לכלול קובץ README שיקרא README.doc, README.docx או README.txt (ולא בשם אחר). הקובץ יכול להיכתב בעברית ובלבד שיכיל את הסעיפים הנדרשים.

קובץ זה יכיל לכל הפחות:

1. כותרת.
2. פרטי הסטודנט: שם מלא כפי שהוא מופיע ברשימות המכללה, ת"ז.
3. הסבר כללי של התרגיל.
4. רשימה של הקבצים שיצרנו, עם הסבר קצר (לרוב לא יותר משורה או שתיים) לגבי תפקיד הקובץ.
5. מבני נתונים עיקריים ותפקידיהם.
6. אלגוריתמים הראויים לציון.
7. תיכון (design): הסבר קצר מהם האובייקטים השונים בתוכנית, מה התפקיד של כל אחד מהם וחלוקת האחריות ביניהם ואיך מתבצעת האינטראקציה בין האובייקטים השונים.
8. באגים ידועים.
9. הערות אחרות.

יש לתמצת ככל שניתן אך לא לוותר על אף חלק. אם אין מה להגיד בנושא מסוים יש להשאיר את הכותרת ומתחתיו פסקה ריקה. **נכתוב ב־README כל דבר שרצוי שהבודק ידע כשהוא בודק את התרגיל.**

אופן ההגשה

הקובץ להגשה: ניתן ליצור בקלות קובץ zip המותאם להגדרות ההגשה המפורטות להלן ישירות מתוך VS, כפי המוסבר תחת הכותרת "יצירת קובץ ZIP להגשה או לגיבוי" בקובץ "הנחיות לשימוש ב־Visual Studio 2022". אנא השתמשו בדרך זו (אחרי שהגדרתם כראוי את שמות הצוות ב־MY_AUTHORS: **נשים את שמות המגישים בתוך המרכאות, נקפיד להפריד בין השם הפרטי ושם המשפחה בעזרת קו תחתי ואם יש יותר ממגיש אחד, נפריד בין השמות השונים בעזרת מקף (מינוס '-')**) וכך תקבלו אוטומטית קובץ zip המותאם להוראות, בלי טעויות שיגררו אחר כך בעיות בבדיקה.

באופן כללי, הדרישה היא ליצור קובץ zip בשם oop1_project-firstname_lastname.zip (או במקרה של הגשה בזוג – oop1_project-firstname1_lastname1-firstname2_lastname2.zip), כשהקובץ כולל את כל קובצי הפרויקט, למעט תיקיות out ו־vs. כל הקבצים יהיו בתוך תיקייה ראשית אחת.

את הקובץ יש להעלות ל־Moodle של הקורס למשימה המתאימה. בכל מקרה, **רק אחד** מהמגישים יגיש את הקובץ ולא שניהם.

הגשה חוזרת: אם מסיבה כלשהי החלטתם להגיש הגשה חוזרת יש לוודא ששם הקובץ זהה לחלוטין לשם הקובץ המקורי. אחרת, אין הבודק אחראי לבדוק את הקובץ האחרון שיוגש.

כל שינוי ממה שמוגדר פה לגבי צורת ההגשה ומבנה ה־README עלול לגרור הורדת נקודות בציון.

מספר הערות:

1. נשים לב לשם הקובץ שאכן יכלול את שמות המגישים.
2. נשים לב לשלוח את תיקיית הפרוייקט כולה, לא רק את קובצי הקוד שהוספנו. תרגיל שלא יכלול את כל הקבצים הנדרשים, לא יתקבל וידרוש הגשה חוזרת (עם כללי האיחור הרגילים).

המלצה כללית: אחרי הכנת הקובץ להגשה, נעתיק אותו לתיקייה חדשה, נחלץ את הקבצים שבתוכו ונבדוק אם ניתן לפתוח את התיקייה הזו ולקמפל את הקוד. הרבה טעויות של שכחת קבצים יכולות להימנע על ידי בדיקה כזו.

בהצלחה!