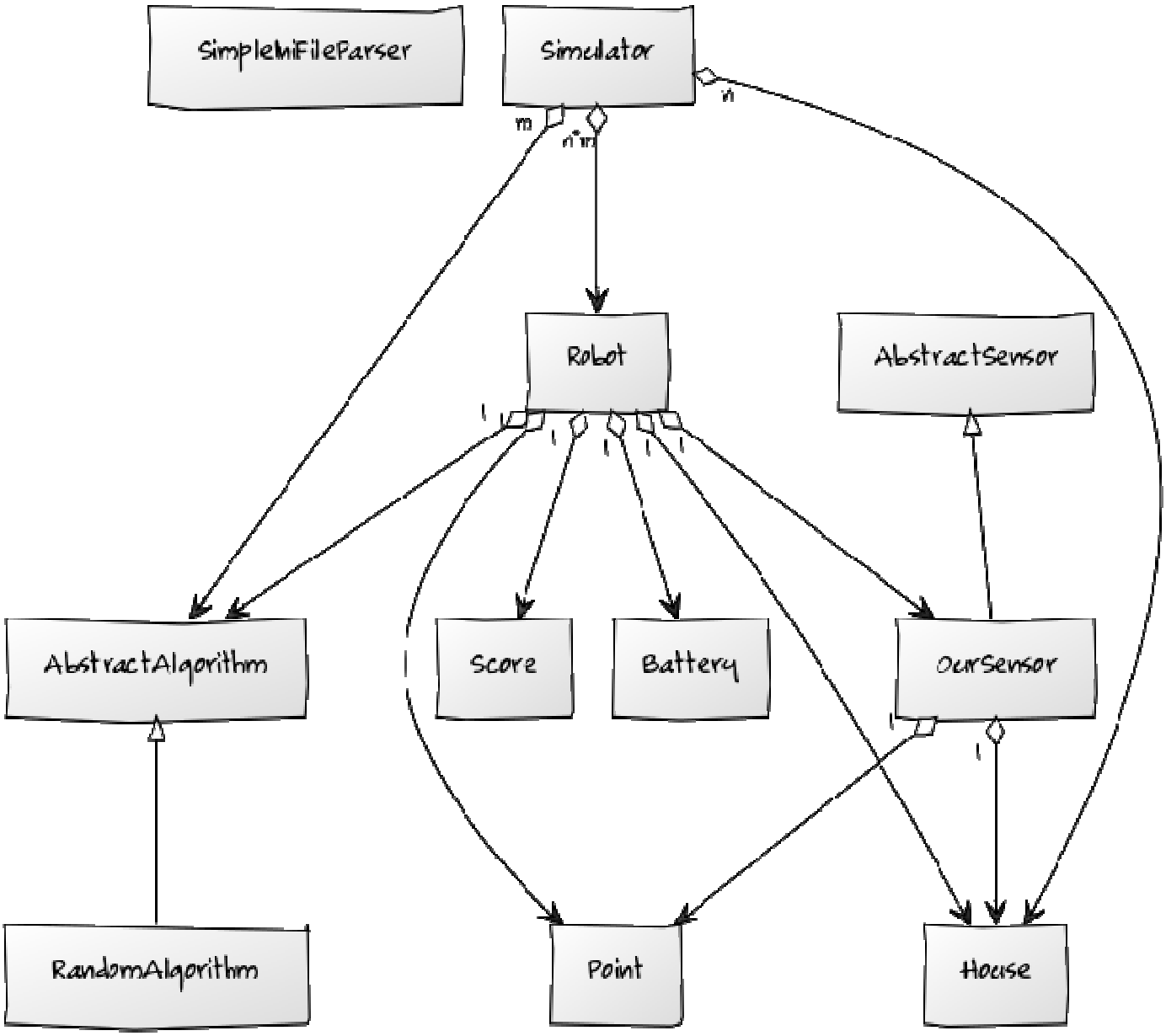
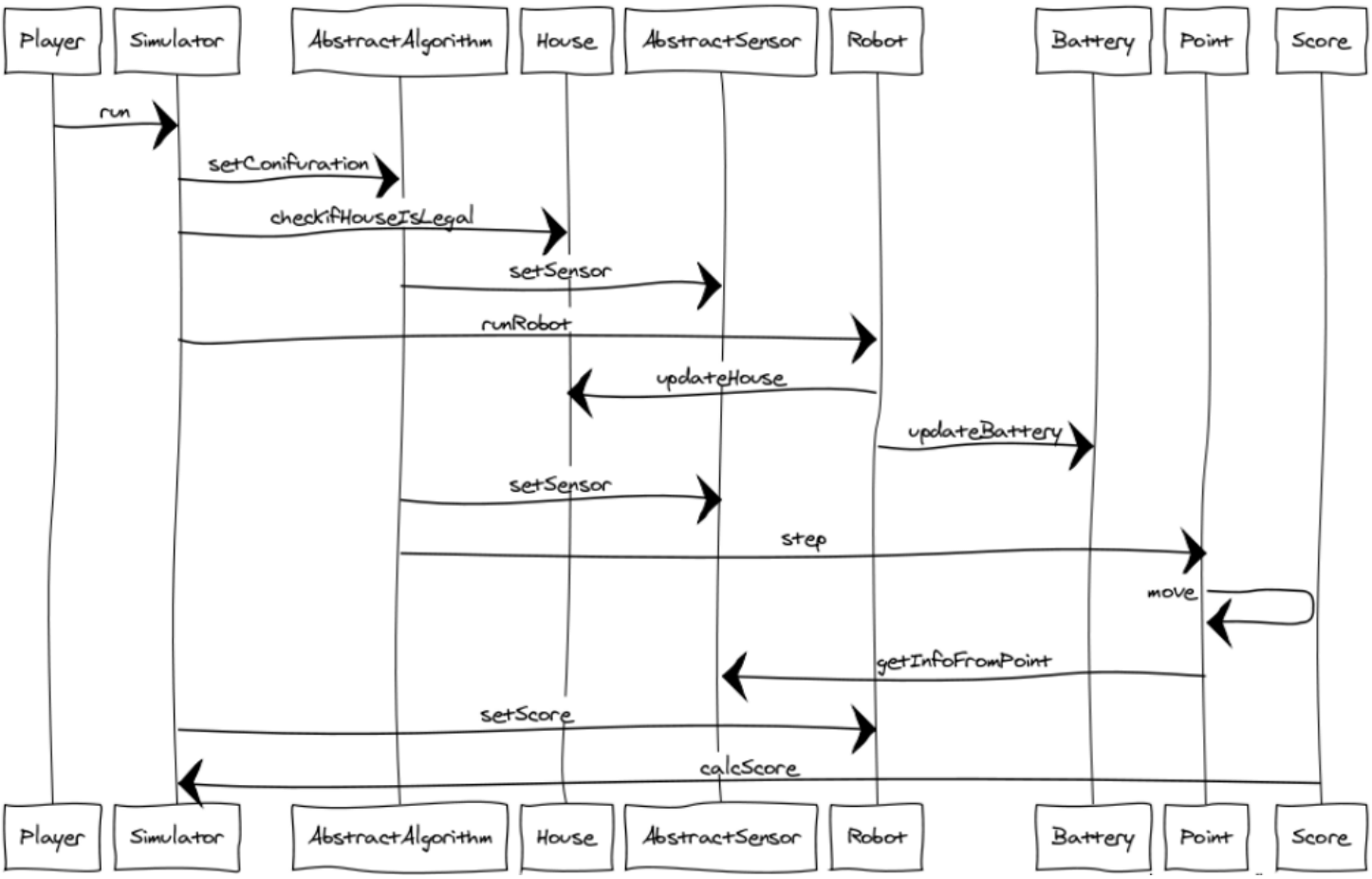
דיאגרמות, והסבר על המימוש של תרגיל 1:

רותם גרפל ת"ז 203246509
מתן לוי ת"ז 304978372

class diagram



sequence-time diagram



המחלקות שיצרנו:

- **Simulator :**

✓ שדות:

```
vector <House*> houses;
vector <AbstractAlgorithm*> algorithms;
map<string, int> config;
```

✓ תפקידה היא לסמלץ את כל הרובוטים בלולאה. הסבר נוסף על המחלקה בהמשך.

- **Robot:**

```
House *house;
AbstractAlgorithm *algo;
OurSensor *sensor;
Point* position;
Battery* battery;
Score score;
bool canRun;
bool brokedDown;
```

✓ לרובוט יש בית, אלגוריתם, חיישן, מיקום, בטרייה, וניקוד. בנוסף שני משתנים בוליאניים:

✓ canRun שהערך הדיפולטי שלו הוא true, אם הרובוט מנצח או שהבטרייה נגמרה באמצע או שהוא התנגש בקיר, הערך מעודכן לfalse.

✓ brokedDoen שהערך הדיפולטי הוא False. כלומר, הערך הדיפולטי הוא שהבית תקין. אם הרובוט מתנגש בקיר/נגמרה הבטרייה באמצע אנחנו יודעים שלא ניתן להריץ את הרובוט בגלל פעילות לא טובה.

✓ משתנים בולאניים אלו עוזרים לנו בלולאה של הסימולטור, לדעת אם ניתן להריץ את הרובוט או לא. בנוסף, BrokedDown עוזר בחישוב הניקוד הסופי.

✓ תפקיד: אחראי על הרצת הסימולציה.

- **OurSensor:**

✓ המימוש שלנו לsensor.

✓ מכיל את השדות:

```
SensorInformation sensorInfo;
House *thisHouse;
Point *currPoint;
```

- **Battery:**

✓ מכיל את השדות:

```
int capacity;
int conRate;
int rachRate;
int currentState;
```

- **House:**

✓ שדות:

```
int R;
int C;
string shortDes;
string longDes;
string* matrix;
```

- **RandomAlgorithm:**

✓ המימוש שלנו לאלגוריתם הנאיבי.

✓ מכיל את השדות:

```
OurSensor* thisSensor;
map<string, int> thisConfig;
```

- **Point:**

✓ מחלקה שעוזרת לדעת את המיקום בבית.

✓ מכילה את השדות:

```
int x;
int y;
```

Score:

✓ מחלקה שמיועדת כדי לחשב את הניקוד.

✓ מכילה את השדות:

```
int position;

int position;
int winnerNumSteps;
int numSteps;
int dirtCollected;
int sumDirtInHouse;
Bool  isBackInDocking;
```

- **SimpleIniFileParser:**

✓ המחלקה שהמתרגל הראה בתרגול. היא מיועדת לקליטת קובץ הקוניפגוריציות והפיכתו ל-map שמכיל את הקונפיגרוציות.

- המחלקות: SensorInformation , Direction, AbstractAlgorithm ,AbstractSensor שהן חלק מההנחיות.

הסבר כללי:

ראשית, ב-main בדקנו האם קיבלנו קלט, או קלט חלקי, או האם לא קיבלנו קלט ועלינו לחפש בתיקייה הנוכחית קבצים שמסתיימים בסיומת house או קבצים שמסתיימים בסיומת ini, בmain אנו דואגים למצוא את כל הקלטים הדרושים לסימולציה.

לאחר שיש לנו את כל הקלטים הדרושים, אנחנו יוצרים וקטור של בתים, וקטור של אלגוריתמים (שכרגע מכניסים לתוכו רק את האלגוריתם הנאיבי/הרנדומלי שדרוש בתרגיל 1). וקובץ הקונפיגורציות.

אנחנו יוצרים אובייקט חדש מסוג סימולטור, ואובייקט זה מאותחל בבנאי שלו עם: המap של קונפיגורציות, וקטור הבתים וקטור האלגוריתמים.

לאחר מכן, קוראים לפונקציה run של הסימולטור. שהיא אחראית על כל התהליך. ראשית, אנחנו עוברים בלולאה על וקטור הבתים ובודקים האם הבית תקין. אם כן, אנחנו עוברים בלולאה על כל האלגוריתמים, ומייצרים רובוט חדש לכל זוג: אלגוריתם, הבית הנוכחי שנמצא תקין בלולאה. עבור כל רובוט מייצרים בטרייה משלו.

לאחר מכן, אנחנו מריצים את כל הרובוטים שהם נמצאים באותו בית (רק עם אלגוריתם שונה) כל אחד צעד אחד (כאשר מספר הצעדים מוגבל ע"י MaxSteps שנתון ע"י map הקוניפגורציות). כשאחד מהרובוטים מנצח, מעדכנים את המיקום שלו ואת הניקוד שלו, ואת מספר הצעדים של הרובוטים האחרים להיות MaxStepsAfterWinner. אם כל הרובוטים בוקטור רובוטים לא יכולים יותר לרוץ, נבדק ע"י הפונקציה allRobotsFinished, הסימולטור יוצא מהלולאה.

בהרצת הרובוט צעד אחד, בפונקציית run של ה-robot, בודקים שהרובוט ניתן להרצה (כלומר, לא ניצח או התנגש בקיר/נגמרה לו הבטריה). אם כן, אנחנו מעדכנים את מיקום הרובוט, בודקים בנשארה עוד בטריה, מגרילים צעד באלגוריתם הרנדומלי ומריצים אותו צעד אחד אם ניתן.

לאחר שהרצנו את כל הרובוטים שמייצגים אלגוריתם שונה, אנחנו מחשבים את הניקוד של כל הרובוטים שעוד לא חישבנו ומדפיסים את הניקוד.