Yair Levi 200945657, Nir Orman 201588902

חלק א' – תיאור המחלקות:

Simulator

המחלקה Simulator הינה המחלקה אשר תנהל את התחרות בין האלגוריתמים השונים.

הסימולטור יבצע:

- טעינת הקונפיגורציה הכללית מתוך קובץ בשם config.ini
- קריאת כל הבתים שמסתיימים ב- house.*, עבור כל בית שנקרא מקובץ, מבצעים יצירה של אובייקט House ומוסיפים אותו לרשימת הבתים. (בתרגיל הראשון תתבצע קריאה של בית בודד).
- קריאת כל האלגוריתמים לזכרון. עבור כל אלגוריתם, מבצעים יצירה של אובייקט NaiveAlgorithm ומוסיפים אותו לרשימת האלגוריתמים. (בתרגיל הראשון יאותחל רק אלגוריתם יחיד).
- הסימולטור יבצע אתחול של אובייקט מטיפוס סימולציה (Simulation) עבור כל זוג של <בית, אלגוריתם>, ועבור הזוג יאותחל אובייקט מטיפוס Sensor. כל הרצה של אלגוריתם מסויים על בית מסויים דורשת שמירה של מצב הבית מבחינת ניקיון ומיקום הרובוט באותו בית בכל רגע ולכן נדרש סנסור נפרד לכל זוג של <בית, אלגוריתם>.
- עבור כל בית, הסימולטור יריץ את כל האלגוריתמים בזה אחר זה בשיטת round Robin בכדי שבכל פעם כל אלגוריתם יבצע רק צעד יחיד על הבית.
- כאשר יש סימולציה שבה הרובוט הגיע לתחנת העגינה והבית נקי, הסימולטור יעדכן את יתר הסימולציות בכמות הצעדים המקסמלית שנותרה להם לבצע.
- כאשר כל הסימולציות סיימו לרוץ (לא משנה מאיזה סיבה כל אחת סיימה לרוץ), יחושב הציון הסופי של כל סימולציה, ויודפס. (בתרגיל הראשון אין צורך לשמור את הציונים כי יש רק סימולציה אחת מפני שיש רק אלרוגיתם יחיד ורק בית יחיד).
- בתום הריצה הסימולטור ינקה את המשאבים שהוא יצר.

Simulation

מחלקה זו מחזיקה בכל הנתונים הנדרשים לצורך ניהול הסימולציה של אלגוריתם על בית מסוים. למשל:

- mStepsCounter מונה של מספר הצעדים שבוצעו בסימולציה
- mScore הציון של הסימולציה
- mDirtCollected כמות האבק שנאספה בבית עד כה
- mCrashedIntoWall האם הסימולציה ביצעה צעד לא חוקי כמו התנגשות בקיר
- מהו ערך המשתנה positionInCompetition עבור סימולציה זו
- mIsRunning האם הסימולציה נמצאת תוך כדי ריצה או שהיא סיימה לרוץ (בעקבות בטריה/ ניצול צעדים מקסימלי/ביצוע צעד לא חוקי)
- mIsBackInDocking האם הרובוט חזר לנקודת העגינה בזמן שהבית נקי
- mIsOutOfBattery האם נגמרה לרובוט הבטרייה
- mInitialDustSumInHouse מהי כמות האבק שהייתה בבית לפני שהרובוט התחיל לנקות אותו.
- מהם הערכים שהגיעו מקובץ הקונפיגוריציות עבור סימולציה זו:
  - mBattreyConsumptionRate
  - mMaxSteps
  - mBatteryCapacity
  - mBatteryRechargeRate

- mMaxStepsAfterWinner

הפונקציה העיקרית של מחלקה זו היא makeSimulationStep אשר גורמת לאלגוריתם לבצע צעד אחד ובודקת מה המצב של הסימולציה לאחר ביצועו. (האם הבית נקי? האם נכנסנו בקיר? האם נגמרה הבטריה? האם הגענו לנקודת עגינה? וכו'). במידה והפוקנציה הזו החזירה true זה אומר שהרובוט הגיע לתחנת העגינה והבית נקי, כלומר הריצה של הסימולציה הסתיימה בהצלחה.

ישנן פוקנציה פומביות נוספות למשל, האם הסימולציה עדיין רצה, ו פונקציה שקובעת את הניקוד של הסימולציה בהתאם לפרמטרים כמו mIsBackInDocking, mCrashedIntoWall, mDirtCollected וכו' לפי הגדרות התרגיל.

$ConfigReader$

מחלקה זו נכתבה ברובה לפי הקוד שניתן לנו בתרגול. מטרתה לגלות את הפרמטרים של הסימולציות לפי קובץ הקונפיגורציות, ולוודא שכל הפרמטרים הנדרשים אכן קיימים בקובץ. במימוש שלנו, במידה ופרמטר כלשהו לא קיים- לא תתבצע ריצה והסימולטור יסיים את ריצתו.

$AbstractAlgorithm$

מחלקה זו היא אבסטרקטית, ובעלת פונקציות וירטואליות שמומשו תחת המחלקה $AlgorithmNaive$.

$AlgorithmNaive$

מחלקה זו יורשת מן המחלקה $AbstractAlgorithm$, וממשת את הפונקציות שבה.

הפונקציה המרכזית היא step() בחרנו לממש אלגוריתם שהוא לא לגמרי רנדומי, אלא כזה שמנקה את הנקודה שבה הוא נמצא עד שאין בה יותר אבק, ואם בנקודה בה הוא נמצא אין אבק, הוא לא יגריל את האפשרות של Direction::Stay מפני שאין טעם להשאר בנקודה בה אין אבק. הפונקציה aboutToFinish מומשה בתרגיל זה באופן ריק מפני שאין לה משמעות עם אלגוריתם יחיד.

$AbstractSensor$

מחלקה זו היא אבסטרקטית, ובעלת פונקציה וירטואלית sense וdestructor וירטואלי שמומשו תחת המחלקה $Sensor$.

$Sensor$

מחלקה זו יורשת מן המחלקה $AbstractSensor$, וממשת את הפונקציות שבה.

במחלקה זו ממומשת sense אשר מחזירהSensorInformation אשר באמצעותו הסימולטור יודע לעקוב על חוקיות צעדי האלגוריתם. בנוסף, היא מאפשר לאלגוריתם עצמו לקבל מידע שישמש אותו בכדי לדעת איזה צעד ביכולתו לבצע.

בנוסף היה הכרח להוסיף לסנסור פונקציה של moveSensor בכדי שנוכל לעדכן אותו במיקומו החדש לאחר שהאלגוריתם בחר לאן להזיז את הרובוט, ולכן בחרנו לשמור גם שדה פרטי שבו אנחנו שומרים את מיקום הסנסור.

הconstructor של הסנסור מאותחל אל תחנת העגינה, מפני שתחילת ריצה של רובוט בבית כלשהו תמיד מתחילה בנקודה זו. לסנסור יש גם מצביע לבית בכדי שהוא יוכל לדווח את המידע העדכני ביותר לגבי בית נתון מסוים אשר פרטיו משתנים תו"כ הריצה (כמות האבק משתנה בבית תו"כ הריצה).

$House$

מחלקה זו מומשה בהתבסס על קוד שניתן לנו בתרגול.

דאגנו לכך שיהיו setters and getters עבור השדות השונים של הבית שאינם צריכים להיות פומביים.

הפונקציה המרכזית במחלקה זו היא fillHouseInfo שמקבלת filePath ואחראית למלא את כל פרטיו של הבית לפי הקובץ הנתון. בנוסף, בכל רגע ניתן לדעת כמו אבק קיים בבית, מהו מיקום תחנת העגינה בבית, לוודא שמבנה הבית הוא חוקי ולהפדיס את הבית.

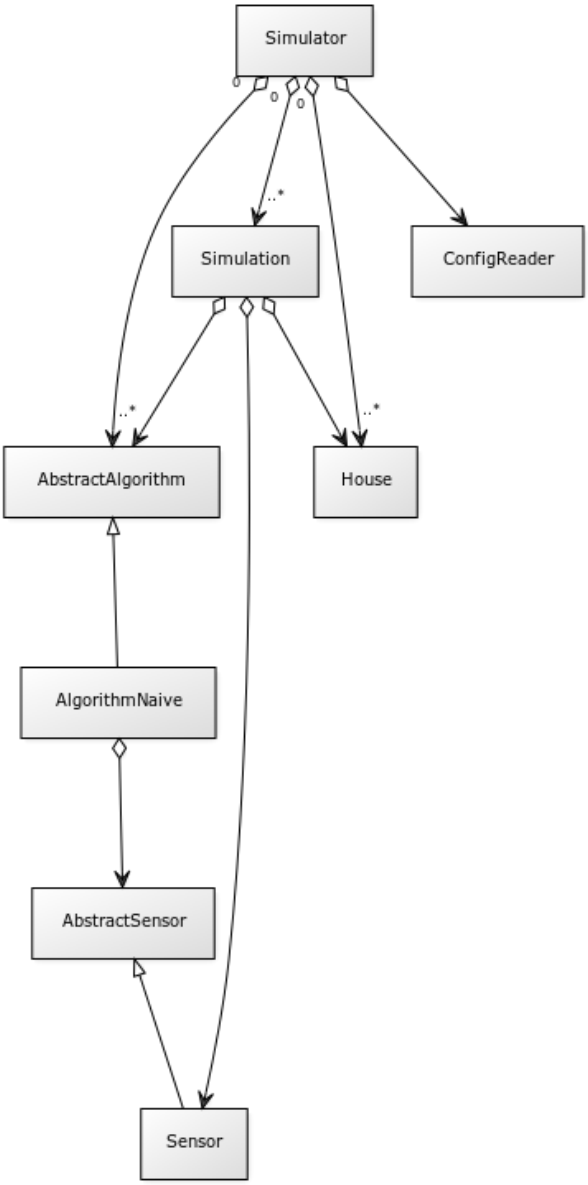
חלק ב'- CLASS DIAGRAM

הסימולטור מחזיק ConfigReader אשר מפרסר את קובץ config.ini עבור הסימלטור ומוודא שכל הפרמטרים הנדרשים אכן קיימים.

הסימולטור מחזיק רשימה של אלגוריתמים **אבסטרקטיים**. הסיבה לכך היא שאנחנו לא מעוניינים שהוא יכיר את המימוש הספציפי של אף אלגוריתם מסויים, ולכן מספיק לו לדעת שכל האלגוריתמים מקיימים את הממשק הבסיסי הנחוץ להרצתם. בתרגיל זה ברשימה יש רק איבר יחיד.

הסימולטור מחזיק רשימה של בתים. זה נחוץ עבור התרגילים הבאים, מפני שבתחילת הריצה של הסימולטור אנו מקבלים תיקיה ומוצאים בה את כל הבתים הקיימים, ומעוניינים לאתחל אובייקט מתאים עבור כל אחד מהם. לכן מתוך מחשבה על התרגילים הבאים, יצרנו רשימה זו. בתרגיל זה ברשימה יש רק איבר יחיד.

בנוסף, בעת הרצת הפונקציה של הסימולטור שנקראת executeAllAlgoOnAllHouses, הסימולטור יוצר באופן דינאמי אובייקטים מטיפוס סימולציה, בכמות ששוה למספר האלגוריתמים. הסימולצית מוחזקות ברשימה. כאשר כל הסימולציות שרצות על בית מסויים מסיימות לרוץ, אנו משחררים את האובייקטים של הסימולציות ויוצרים סימולציות חדשות עבור הבית הבא, באותה כמות – לפי מספר האלגוריתמים (כמובן שזה מתבצע לאחר שמירת תוצאת הניקוד של כל סימולציה). בתרגיל הזה ישנה רק סימולציה אחת ברשימת הסימולציות, מפני שיש רק אלגוריתם אחד ורק בית אחד.



כל סימולציה מחזיקה באובייקט בית יחיד. זאת בכדי לעדכן אותו לפי התקדמות הרובוט בניקיון.

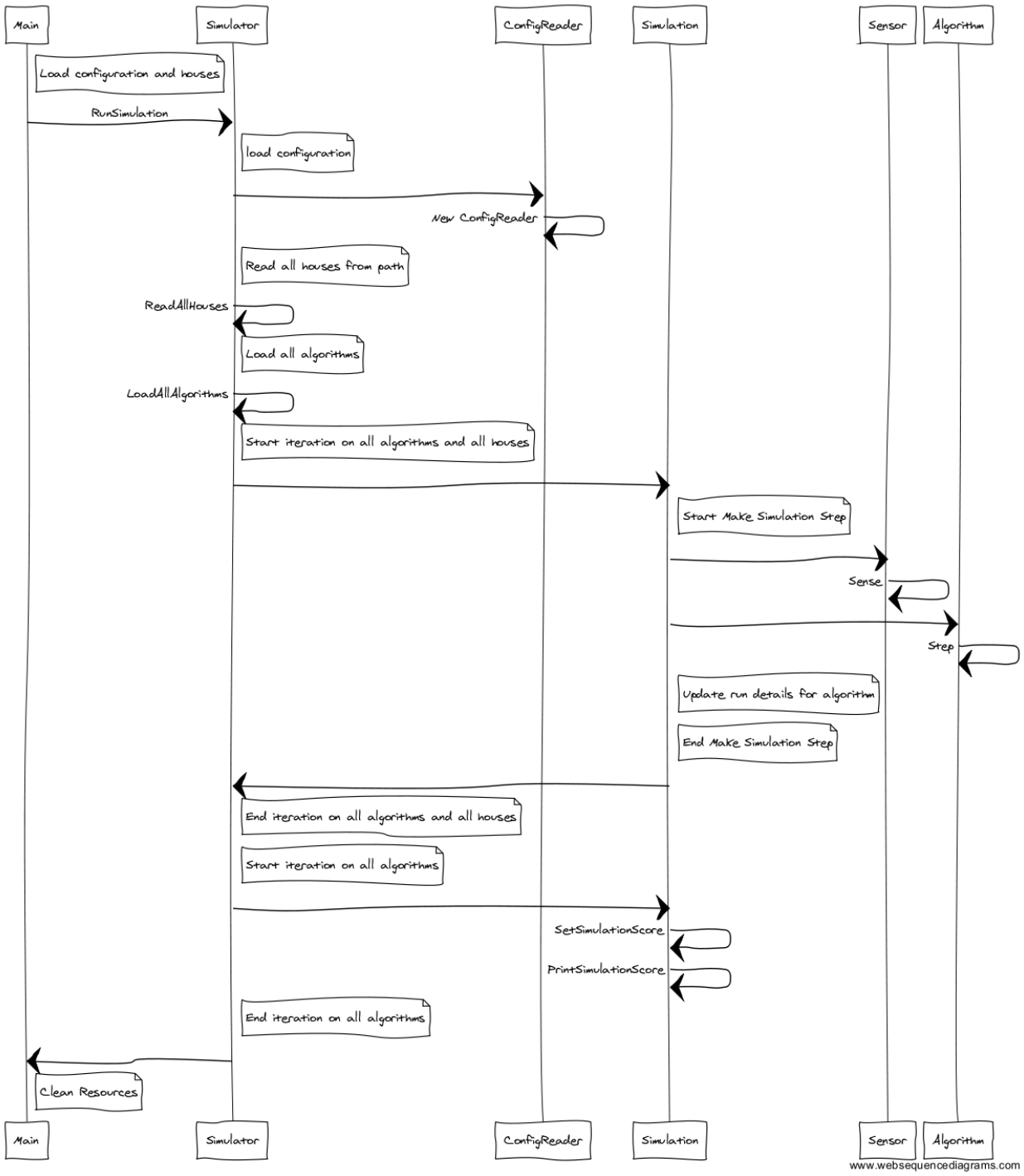
כל סימולציה מחזיקה בנוסף אלגוריתם יחיד, מפני שלאלגוריתם יש סנסור שהוא ספציפי לריצה של האלגוריתם על בית מסויים.

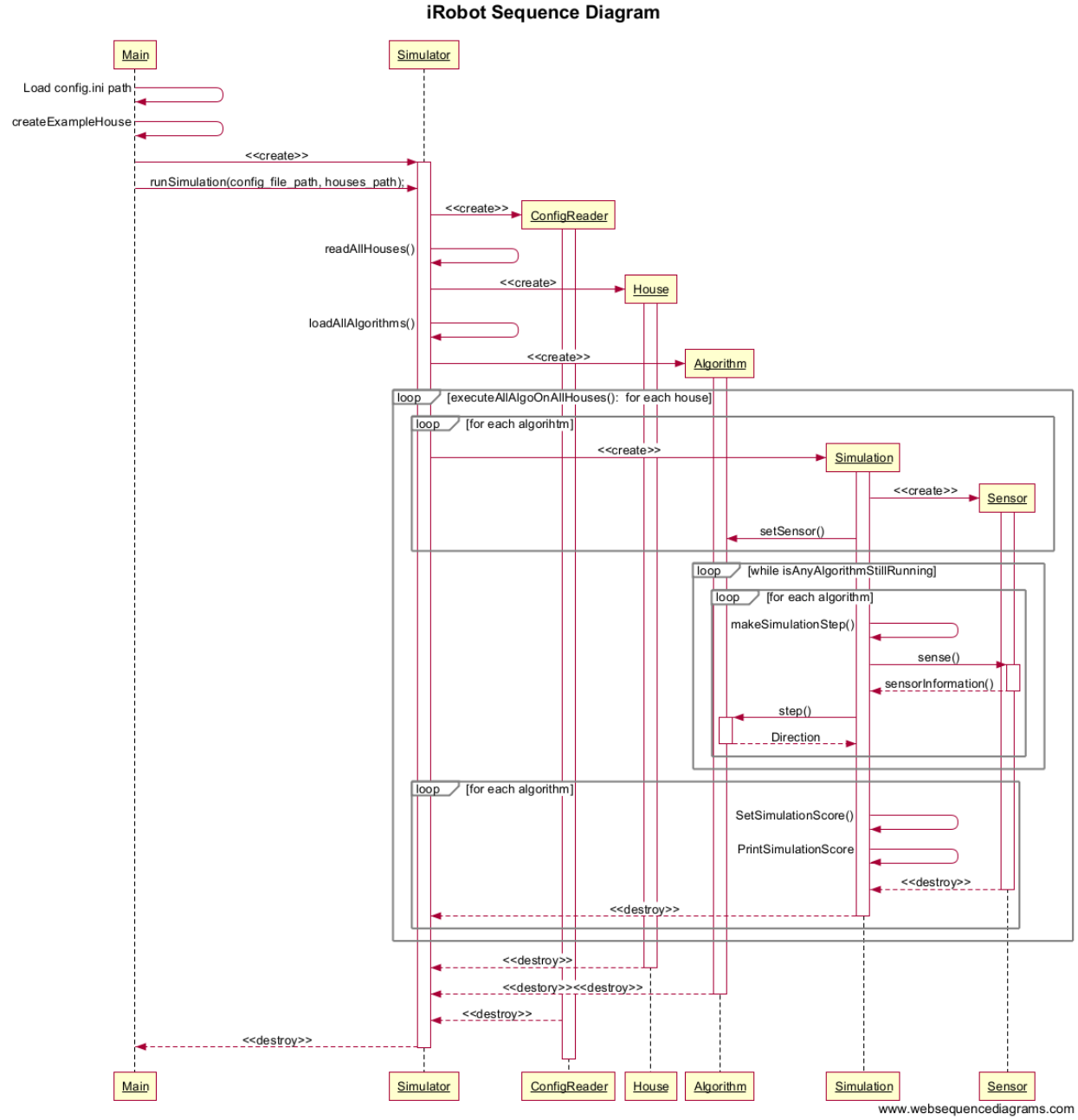
בנוסף, הסימולציה מחזיקה סנסור משל עצמה על מנת להיות מסוגלת לעקוב אחר התקדמות הניקיון כפי שמתבצע ע"י האלגוריתם, ולדעת האם האלגוריתם ביצע צעד שאינו חוקי ולטפל במקרה בהתאם. נשים לב שהסימולציה מחזיקה בסנסור שהוא לא אבסטרקטי כי היא חייבת להצליח לשנות את המיקום שלו ואין פונקציה כזו לסנסור האבסטרקטי.

בניגוד לכך, האלגוריתם הנאיבי מחזיק בסנסור שהוא אבסטרקטי מפני שיתכן שהוא יקבל סנסור במימוש כלשהו, וכל מה שמובטח לגביו הוא שמתקיים הממשק של abstractSensor. כאשר אנו מעבירים לו פרמטר של סנסור בתור reference זה אומנם Sensor שאינו אבסטרקטי, שאנחנו יצרנו, אבל זה רק מפני שלא ניתן לייצר אובייקט מסוג המחלקה האבסטרקטית.

חלק ג' - SEQUENCE DIAGRAM

השתמשנו בדיאגרמה הבאה בכדי לבצע תכנון של הקוד והמחלקות שבו. העמוד הבא נימצא תיאור פורמלי יותר של רצף הקריאות לפונקציה כפי שכתבנו לאחר המימוש הסופי של הקוד.




iRobot Sequence Diagram
Main
Simulator
Load config.ini path
createExampleHouse
<<create>>
runSimulation(config_file_path, houses_path);
<<create>>
ConfigReader
readAllHouses()
<<create>
House
loadAllAlgorithms()
<<create>>
Algorithm
loop
[executeAllAlgoOnAllHouses(): for each house]
loop
[for each algorihtm]
<<create>>
Simulation
<<create>>
Sensor
setSensor()
loop
[while isAnyAlgorithmStillRunning]
loop
[for each algorithm]
makeSimulationStep()
sense()
sensorInformation()
step()
Direction
loop
[for each algorithm]
SetSimulationScore()
PrintSimulationScore
<<destroy>>
<<destroy>>
<<destroy>>
<<destory>><<destroy>>
<<destroy>>
<<destroy>>
Main
Simulator
ConfigReader
House
Algorithm
Simulation
Sensor
www.websequencediagrams.com