תרגיל 2 חלק 2

בתרגיל זה נרחיב את הפונקציונליות שמימשנו בתרגיל 2.

ניקח את המשחק שמימשנו הזה ונוסיף לו פונקציונליות חדשה:

1. נוסיף למשחק לפחות שני שחקנים אוטומטיים בשם Clever Mrs. ו־Mr. Whatever. סוג השחקן השלישי כבר קיים: השחקן האנושי.
2. עם עליית המשחק, יבחר המשתמש שני סוגי שחקנים. השחקנים האלה יכולים להיות מאותו סוג (שחקן אנושי נגד שחקן אנושי או שחקן אוטומטי נגד שחקן אוטומטי) או מסוגים שונים.
3. השחקנים שנבחרו יערכו טורניר המורכב ממספר סיבובים שיגדיר המשתמש. בכל סיבוב יחליפו השחקנים תפקיד כך שישחקו לסירוגין בסימנים איקס ועיגול, ולוח התוצאות יציג כמה סיבובים ניצח שחקן 1, כמה סיבובים ניצח שחקן 2 וכמה סיבובים הסתיימו בתיקו.
4. כמו כן, יבחר המשתמש משתנה שלישי: כיצד ירונדר הלוח. האפשרויות הן:

   א) בחלון שורת הפקודה (כמו מקודם)

   ב) הלוח לא ירונדר. האפשרות האחרונה שימושית במיוחד כשרוצים להריץ טורניר של הרבה סיבובים בין שחקנים אוטומטיים.
5. התרגיל כולל אתגר: בעוד שהשחקן Whatever Mr. יבחר אסטרטגיה באופן אקראי, אתם תבחרו את האסטרטגיה עבור השחקנית Clever Mrs. ויהיה עליה לנצח בשיעור גבוה מאוד של הסיבובים נגד Whatever. Mr. זה אולי נשמע כמו משהו מורכב ומסובך, אבל למעשה לא צריך הרבה כדי להערים על Whatever. Mr.

6. אתגר נוסף שתצטרכו לממש הוא שחקן שמשחק טוב יותר מ Mrs. Clever שיקרא Mr. Snartypamts

ליתר דיוק, משתמש או משתמשת שמריצים את המשחק שלנו יקלידו את הפקודה הבא בחלון שורת הפקודה:

```
java Tournament [round count] [render target: console/none] [player: human/clever/whatever/...] X 2
```

לדוגמה: כדי לשחק טורניר של 10,000 סיבובים בין Mrs. Clever ל־ Mr. Whatever מבלי להדפיס את לוחות המשחק האלה בחלון שורת הפקודה, הפקודה היא:

```
whatever clever none 10000 Tournament java
```

כדי לשחק 3 משחקים בין שחקן אוטומטי למשתמש, הפקודה היא:

```
human clever console 3 Tournament java
```

מה יקרה אם תזינו את הערך none עבור רינדור הלוח אבל תגדירו גם שחקן אנושי?

תקבלו בדיוק את מה שביקשתם: הלוח לא יודפס על המסך אבל כשיגיע תורו של המשתמש האנושי הוא או היא עדיין יתבקשו לבחור משבצת בלוח המשחק.

הפרמטרים ששולטים על מבנה הלוח (גודלו והרצף הנדרש לניצחון) יוגדרו כבעלי ערכים דיפולטיבים במחלקה)WIN_STREAKBoard יוגדר להיות 4 ו SIZE יוגדר להיות 6).

נפרט קצת על מבנה המחלקות במשחק, תוך השוואה למבנה שניתן בתרגיל 2.

תפקידה יהיה להריץ את כל הסיבובים/משחקים דרך המחלקה .Game

בתורה, תמשיך המחלקה Game להשתמש במחלקה Board.
חוץ מהעובדה שנוציא את main מהקובץ Game.java, המחלקה Game
לא תשתנה כלל. למעשה, המחלקות Tournament ו־Game שאחראיות להריץ
את המשחק יישארו ללא שינוי אפילו כשנוסיף סוגי שחקנים נוספים.
אף־על־פי־כן, יש מקום אחד (ורק אחד) שישתנה בכל פעם שנוסיף סוג שחקן חדש.
הסיבה לכך היא שאנחנומקפידים על 'עקרון האחריות הבודדת'
(Single Choice Principle). המקום הזה הוא המחלקה
PlayerFactory שאחראית על יצירת השחקנים.
לאחר מכן, ניצור את המחלקות עבור השחקנים CleverPlayer ו־WhateverPlayer.
שמה של המחלקה שנקראה בעבר Player ישתנה והיא תיקרא עכשיו
HumanPlayer.
עבור שלושת סוגי השחקנים האלה, ועבור כל סוג שחקן נוסף, ימומש ממשק חדש: Player.
גם המחלקה Renderer הישנה תשאר ללא שינוי(חוץ ממה שאמרנו), אבל שמה ישתנה
ל־ConsoleRenderer.
כמו כן, ימומש ממשק חדש בשם Renderer.
המחלקה השנייה שתשתמש בממשק Renderer למעשה לא תצייר כלום על המסך
(כך ששחקנים אוטומטיים יוכלו להתחרות זה בזה מבלי שלוחות הממשק ימלאו
את שורות הפקודה), אבל היא עדיין צורה של רינדור.
היא תקבל את השם VoidRenderer (מכיוון שהיא מרנדרת void, כלומר כלום)
והיא תממש את אותו ממשק בדיוק. מפעל נוסף בשם RendererFactory יהיה אחראי
על יצירת ממשק ה־Renderer המתאים. זאת דרך פשוטה ואלגנטית להשאיר לעצמנו את
האפשרות להוסיף בעתיד למשחק אפשרויות רינדור נוספות, כשכל מה שצריך הוא לשנות
את המפעל ולא שום חלק אחר בקוד.

נסכם את השינויים הנדרשים בקוד הקיים:

1. שינוי שם המחלקות Player ו־Renderer לשמות HumanPlayer ו־ConsoleRenderer, בהתאמה.
2. המחלקות הנ"ל צריכות לממש את הממשקים Player ו־Renderer בהתאמה.
3. חילוץ השיטה *main* מהמחלקה Game. השיטה *main* החדשה תהיה כעת חלק מהמחלקה החדשה Tournament.

הממשקים Player ו־Renderer

החל מהפרק הבא, שמה של המחלקה שכרגע נקראת Player ישתנה ל־HumanPlayer וכך יתייחס אליה המדריך. באופן דומה, שמה של המחלקה שכרגע נקראת Renderer ישתנה ל־ConsoleRenderer וכך יתייחס אליה המדריך. השמות Player ו־Renderer ישמשו עבור ממשקים חדשים.

הממשק Player מבטא שחקן איקס עיגול כללי. מחלקות אחרות שמצביעות אל הממשק Player לא ידעו מהו סוג השחקן שאליו הן מצביעות, רק שהוא מממש את ה־API של הממשק Player וכנ"ל לגבי הממשק Renderer.

ממשק Player

בואו נעבור על הקוד של המחלקה Player שכבר קיימת.

מהו ה־API שלה? אם הקפדתם על עקרון האבסטרקציה בעיצוב ה־API, הוא לא צריך להתאים רק לשחקן אנושי, אלא לכל סוגי השחקנים. מכיוון שתפקידו של ממשק Player החדש הוא להציג רשימה של השיטות הצפויות עבור שחקן גנרי, ה־API שלו צריך להיות כמעט זהה(כאן לא יהיה בנאי).

שנו את שם המחלקה והקובץ מ־Player ל־HumanPlayer. הוסיפו במקומה ממשק

Player חדש עם ה־API הנכון. לא בטוחים כיצד לממש ממשק? העזרו בסעיף שעוסק בממשקים שבפרק טיפים למימוש. כתבו קוד מתאים כך שהמחלקה HumanPlayer תממש את הממשק Player.

נו את שם המחלקה והקובץ מ־Player ל־HumanPlayer. הוסיפו במקומה ממשק Player חדש עם ה־API הנכון. לא בטוחים כיצד לממש ממשק? העזרו בסעיף שעוסק בממשקים שבפרק טיפים למימוש. כתבו קוד מתאים כך שהמחלקה HumanPlayer תממש את הממשק Player

ממשק Renderer

נחזור על התהליך עבור המחלקות Renderer ו־ConsoleRenderer. חשבו על הקוד של המחלקה Game וכיצד הוא משתמש במחלקת Renderer הישנה; כיצד לדעתם צריך להיראות ה־API של הממשק החדש?

שנו את השם של המחלקה Renderer ל־ConsoleRenderer והוסיפו את הממשק Renderer החדש.

שימו לב: כשאתם משנים את השם של המחלקה Renderer ל־ConsoleRenderer קיים גם בנאי שאת שמו צריך לשנות.

סוגי הטיפוסים הבאים כבר מוכנים:

1. Game
2. Board
3. Player
4. Renderer
5. HumanPlayer
6. ConsoleRenderer

ומכאן שטיפוסי הליבה שחסרים לנו הם:

1. Tournament
2. PlayerFactory
3. RendererFactory

בואו נתחיל עם Tournament. צרו קובץ והעבירו את השיטה main מהמחלקה Game למחלקה Tournament. לעת עתה, אל תדאגו לגבי השיטות האחרות במחלקה - המחלקה Tournament צריכה לכלול כרגע רק את שיטת ה־main מהתרגיל הקודם.

המפעל PlayerFactory אחראי למפות את המחרוזת, כפי שהועברה לו בשורת הפקודה (כלומר "clever" / "whatever" / "human"), לאובייקט שחקן ממשי. אין סיבה להשתמש ביותר משיטה אחת(בנוסף לבנאי), טיפוס הקלט של שיטה זו יהיה string, טיפוס הפלט שלה יהיה ממשק Player. לשיטה זו נקרא buildPlayer באופן דומה נטפל בRendererFactory רק שהוא יצפה לקבל את המחרוזות console, none.

ה-API של Tournament

נגדיר בבירור את תפקיד המחלקה. המחלקה Tournament מבצעת סדרה של משחקי איקס עיגול (סיבובים) בין שחקנים מסוימים ובממשק רינדור מסוים, כשבין המשחקים מחליפים השחקנים תפקידים (אם במשחק הראשון שחקן 1 שיחק את X, הרי שבמשחק השני הוא ישחק את O וחוזר חלילה). בסיום כל משחק, מודפסת על המסך התוצאה העדכנית. התוצאה העדכנית מורכבת ממספר הניצחונות של שחקן 1, מספר הניצחונות של שחקן 2 ומספר הפעמים שבהן המשחק הסתיים בתיקו.

כדי לבצע את תפקידה, קרוב לוודאי המחלקה Tournament זקוקה רק לשיטה אחת(מלבד הבנאי) כדי לשחק טורניר בין שני שחקנים . נקרא לשיטה הזאת playTournament שיטה זו לא תקבל ולא תחזיר כלום.

חתימת הבנאי של מחלקה זו היא

Tournament(int rounds, Renderer renderer, Player[] players)

כאשר Player[] מערך המכיל שני שחקנים.

.

Whatever Mr. – התוסף הראשון

Whatever Mr. לא אוהב משחקי חשיבה. כשמגיע תורו, הוא משחק באופן אקראי. עם זאת, מאוד חשובה לו אחידות. חשוב לו שלכל המהלכים האפשריים תהיה הסתברות זהה אתם צריכים לחשוב על דרך פשוטה מאוד להשיג את זה מבלי להסתבך יותר מדי.

המחלקה VoidRenderer

לא תתקשו כאן. המחלקה VoidRenderer צריכה לממש את הממשק Renderer, אבל במימוש ריק. כלומר לא להדפיס דבר.

- **Mrs.Clever**

תכננו אסטרטגיה חכמה יותר מהרנדומית, אתם נדרשים לנצח את היריב הרנדומי ברוב הפעמים.

- **Mr. Snartypamts**

תכננו אסטרטגיה שמנצחת את השחקן החכם ברוב הפעמים.

הבדלים מהתרגיל באתר:

1 **חתימת הבנאי של Tournament שונה**

2 **חובת מימוש ל Mr. Snartypamts**

הנחיות :

עליכם להגיש קובץ jar בשם ex2_2.jar המכיל את הקבצים שפורטו לאורך המדריך, כלומר:

2 Player

3 PlayerFactory

4 WhateverPlayer

5 CleverPlayer

6 HumanPlayer

7 SnartypamtsPlayer

8 Renderer

9 ConsoleRenderer

10 VoidRenderer

11 RendererFactory

12 Board

13 Game

14 Tournament

כמו כן הגישו קובץ README (ללא סיומת) –

בשורה הראשונה בקובץ זה יופיע שם המשתמש שלכם, בשורה השנייה יופיע מספר תעודת הזהות שלכם, השורה השלישית תהיה ריקה.

אתם מתבקשים לפרט בקובץ על בחירת האסטרטגיה של השחקן החכם ועל אופן המימוש של השחקן הטיפש.

כמו כן, הריצו טורניר בן 500 משחקים בין כל אחד מסוגי השחקנים וכתבו בקובץ בכמה משחקים ניצח כל אחד.

הריצו טורניר בן 10000 משחקים בין שני השחקנים הרנדומים וכתבו כמה פעמים כל אחד ניצח(שימו לב, מבחן זה נועד לוודא שהאפליקציה שלכם באמת רנדומית).

למען הסר ספק האסטרטגיה של השחקנים החכמים אמורה להתאים לכל גודל של לוח ולכל רצף.