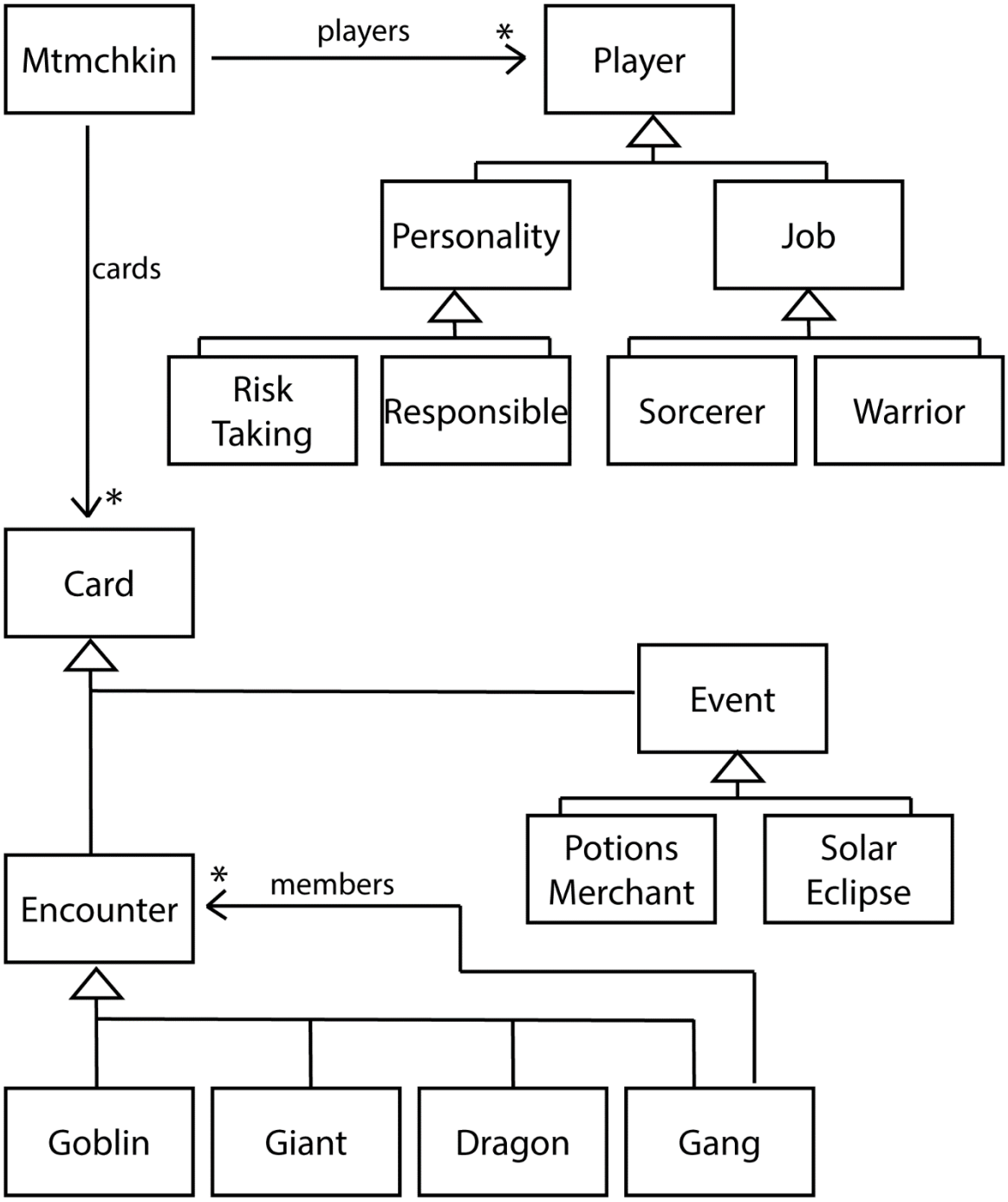
## חלק יבש

סעיף 1)



## סעיף 2)

השתמשנו בDesign Pattern משני סוגים, ניתן גם לראות אותם בדיאגרמת הUML מסעיף הקודם.
הסוג הראשון הוא Strategy, השתמשנו בו במחלקת Player עבור הpersonality וגם עבור Jobs.
הסוג השני הוא composite, שמשומש במחלקת הMtmchkin וגם במחלקת הGang.

## סעיף 3)

ניתן לעשות זאת בכמה דרכים, נבחר ונסביר את השיטה הבאה:

שינויים בJobs:

- תחילה נוסיף Job חדש בשם הNinja שיורש מJobs ואין צורך לשנות את הפונקציות שכבר קיימות חוץ מהgetJob כך שתחזיר את שמו (Ninja).
  לאחר מכן נוסיף מיתודה חדשה escape לJobs שמחזירה true אם הplayer מתחמק מהקרב כך שבJob הדיפולטי הפונקציה תמיד תחזיר שלא (false) אך עבור הninja נעשה override כך שהיא תבדוק אם הוא צריך להתחמק או לא לפי התנאים הנתונים.

שינויים בPlayer:

- נדרש לשנות הפונקציה setJob כך שתקבל Ninja בתנאים שלה.

שינויים בCard:

- שינוי דרוש הוא במחלקה Encounter בפונקציה ()playCard נוסיף if בהתחלה שתבדוק אם הplayer יתחמק או לא, אם כן אז נסיים את הרצת הפונקציה ללא השפעה על הplayer ונחזיר תוצאת הencounter, אחרת נמשיך בפונקציה כרגיל.

שינויים בFiles:

- נצטרך להוסיף את הNinja גם בקלט האפשרי לjobs בפונקציה playerFiles.

## סעיף 4)

ניתן לממש את זה בקלות במערכת שלנו מכיוון שהjob הוא שדה אצל הplayer שממומש ע"י strategy pattern לכן ניתן לשנותו בקלות. כדי לעשות את זה:
נוסיף מיתודה עבור הplayer שתבחר job רנדומלי ואז בעזרת הפונקציה ()setJob שקיימת אצלנו נשנה את הJob. נוסיף גם את הDivineInspiration שיורש מהEvent כך שה()playCard שלו קורא לפונקציה שנזכרה לפני.
וגם עבור ההדפסות יש צורך לשנות את ה()getDescription בהתאם ולהוסיף את ההדפסות של הEvent לתוך ה()playCard. ועבור הקלט בקובץ Files.cpp נוסיף אפשרות לקבל את הקלף הזה.