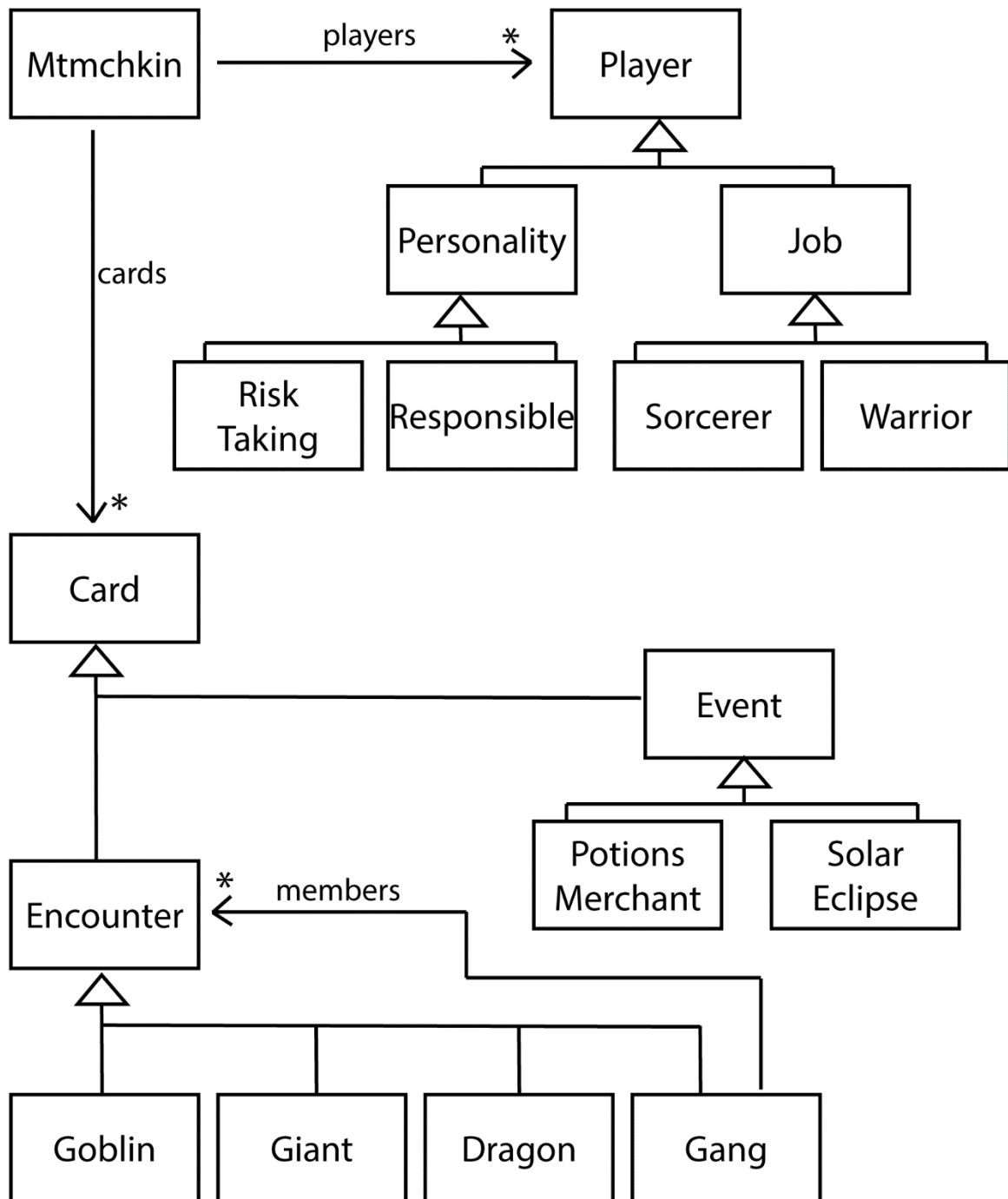


חלק יבש

סעיף 1)



סעיף 2)

השתמשנו בDesign Pattern משני סוגים, ניתן גם לראות אותם בדיאגרמת UML מסעיף הקודם. הסוג הראשון הוא Strategy, השתמשנו בו במחלקת Player עבור personality וגם עבור Jobs. הסוג השני הוא composite, שמשמש במחלקת Mtmchkin וגם במחלקת Gang.

סעיף 3)

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

שינויים בJobs:

- תחילה נוסיף Job חדש בשם Ninja שיוורש מJobs ואין צורך לשנות את הפונקציות שכבר קיימות חוץ מהJob getJob כך שתחזיר את שמו (Ninja).
לאחר מכן נוסיף מיתודה חדשה escape לJobs שמחזירה true אם player מתחמק מהקרב כך שבJob הדיפולטי הפונקציה תמיד תחזיר שלא (false) אך עבור ninja 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() נוסף if בהתחלה שתבדוק אם player יתחמק או לא, אם כן אז נסיים את הרצת הפונקציה ללא השפעה על player ונחזיר תוצאת encounter, אחרת נמשיך בפונקציה כרגיל.

סעיף 4)

ניתן לממש את זה בקלות במערכת שלנו מכיוון שהjob הוא שדה אצל player שממומש ע"י strategy pattern לכן ניתן לשנותו בקלות. כדי לעשות את זה:
נוסיף מיתודה עבור player שתבחר job רנדומלי ואז בעזרת הפונקציה setJob() שקיימת אצלנו נשנה את הJob. נוסיף גם את הDivineInspiration שיוורש מEvent כך שהplayCard() שלו קורא לפונקציה שנזכרה לפני.
וגם עבור ההדפסות יש צורך לשנות את הgetDescription() בהתאם ולהוסיף את ההדפסות של Event לתוך הplayCard().