**Summary**

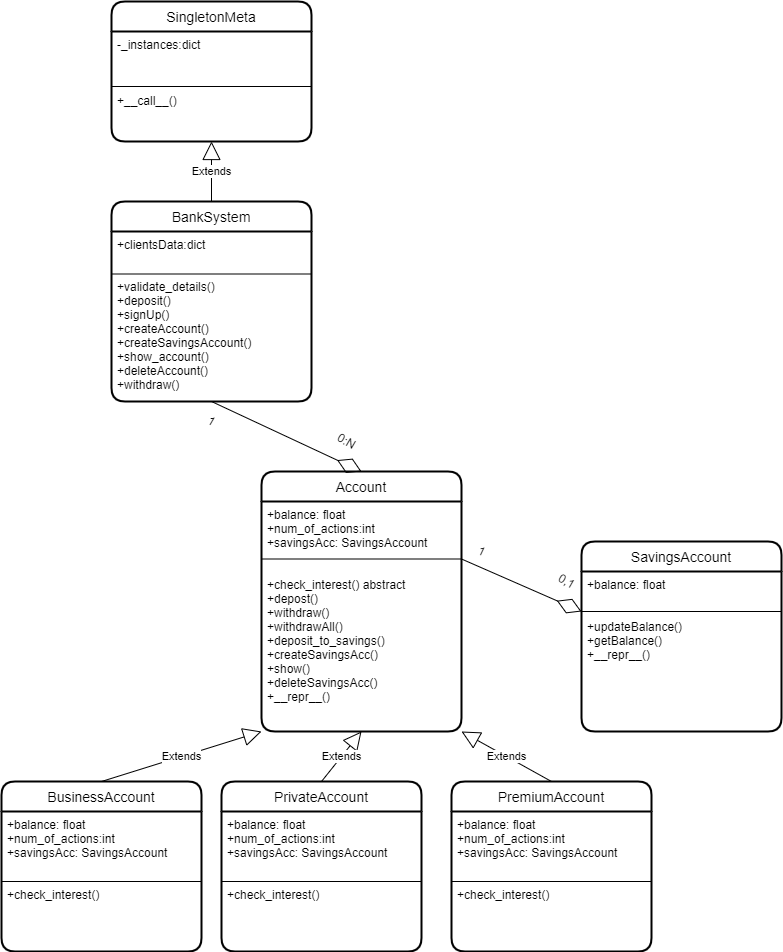
**מבנה המודולים:**

בתוכנית בחרנו לבנות 4 מודולים(קבצים):

1. Main - מכיל מימוש ל-CLI, הממשק שמנהל את התוכנית
2. BankSystem – מכיל מחלקת הבנק השדות והפונקציונליות הנדרשות
3. Account – מחלקת חשבון אבסטרקטית, ועוד שלוש מחלקות של סוגי החשבונות שיורשות את מחלקת חשבון
4. SavingsAccount – מחלקת חשבון חסכון שמכילה שדות ופונקציות הנדרשות

* חילקנו את המודולים בצורה כך שלכל מודול יש גישה רק לקבצים(מחלקות) שקיים ביניהם קשר מסוים(לפי השרטוט)
* פירוט על איזה שדות/פונקציות יש לכל מחלקה מופיע באיור(UML) למטה
* לא בחרנו לממש מחלקת לקוח כיוון שזה מיותר במבנה שבחרנו, הסבר על החלופה לכך מופיע למטה בסעיף של המחלקות

**דיאגרמת UML:**



**מחלקות:**

**מחלקת מערכת הבנק**: היא המחלקה הגדולה ביותר שהיינו צריכים לממש כיוון שהיא מייצגת ממשק לכל הפונקציונליות שה-CLI יכול להשתמש בהם, והיא גם מייצגת את בסיס הנתונים שמכיל את כל המידע עבור הלקוחות והחשבונות.

**מחלקת חשבון בנק:** היא המחלקה הכי חשובה במבנה שבחרנו לממש כי רוב הפונקציונליות שהמערכת תומכת בהם קשורות לחשבונות הבנק, בנוסף לכך, היינו צריכים לבנות 3 מחלקות היורשות ממחלקת חשבון האב כך שכל מחלקה מייצגת סוג חשבון אחר ומממשת פונקציונליות אחרת לחישוב הריבית.

**מחלקת חשבון חסכון:** בחרנו לממש את המחלקה הזאת כמחלקה נפרדת ממחלקת החשבון הרגיל כי חשבון חסכון הוא לא **סוג של** חשבון, אלא הוא חלק מחשבון רגיל, כך שחשבון יכול להכיל או לא להכיל חשבון חסכון(Abstract Factory).

**תבניות עיצוב:**

**Singleton**: מערכת הבנק מייצגת סינגלטון, כך שקיים מופע אחד מהמחלקה ואין אפשרות ליצור מופע אחר, בחרנו לממש אותה בצורה הזאת כי מדובר במערכת בנק אחת.

**Abstract Factory:** השתמשנו בתבנית העיצוב הזאת במחלקת חשבון, כך שמחלקת החשבון היא ה-Factory של מחלקת חשבון חסכון, הקשר ביניהם הוא קשר הכלה, ומחלקת חשבון אחראית על בניית מופעים של חשבון חסכון.

**Factory Method:** השתמשנו בתבנית העיצוב במודול Account, כך שלמחלקת החשבון האבסטרקטית קיימים 3 בנים שיורשים ממנה, כל מחלקות הבנים יורשים את כל השדות\פונקציונליות של מחלקת האב חוץ מפונקציה אחת שכל אחד מהבנים ממש אותה לפי דרישתו(check\_interest).