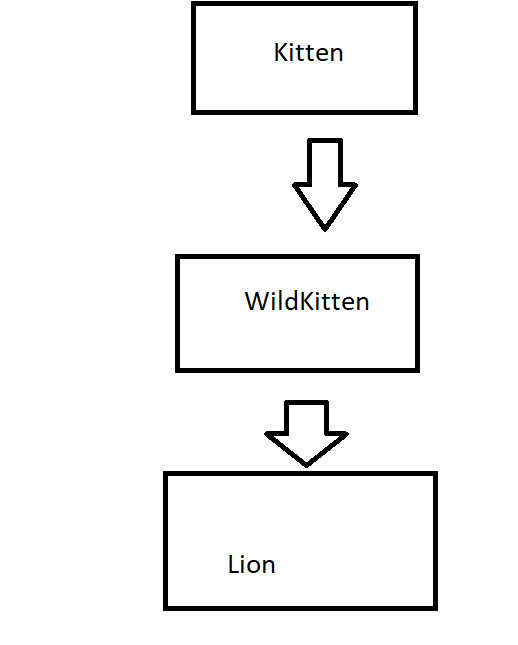
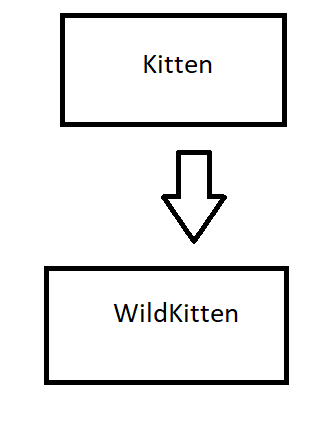
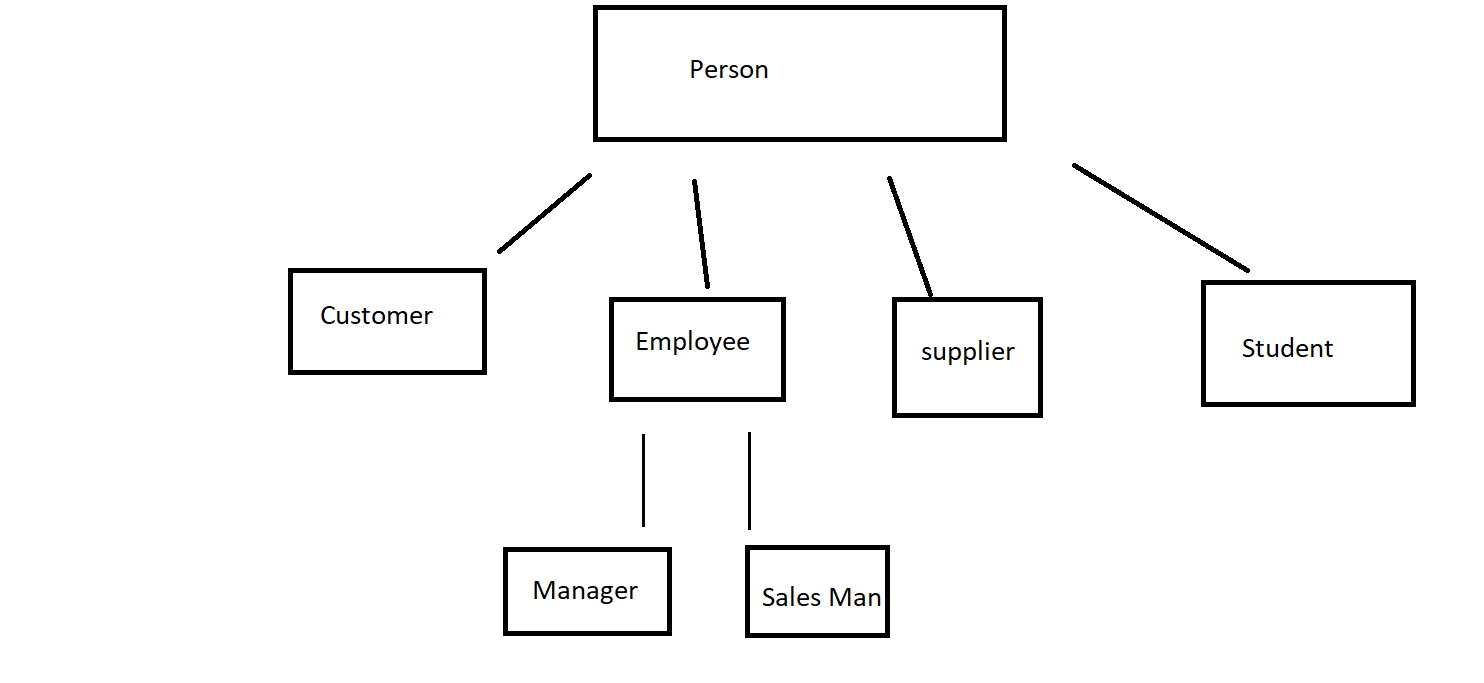
**DOC­S – Advanced JavaScript 2023-02-06**

TypeScript

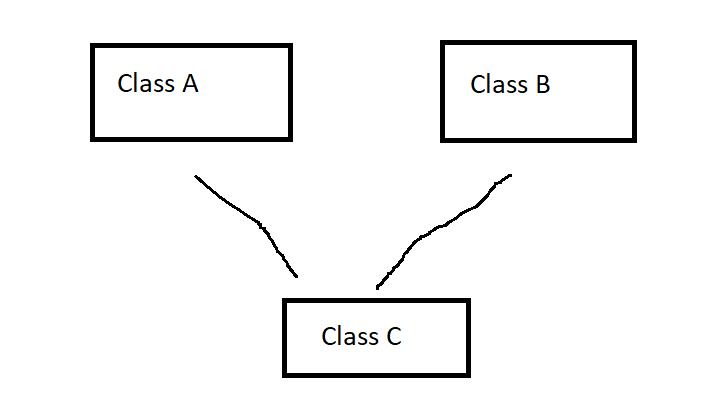
* כאשר אנחנו יורשים מחלקת אב ( SUPER ), אנו מקבלים גם את השדות וגם את המתודות וגם את הקונסטרקטור.
* לראיה – אם נרצה לבדוק כיצד זה פועל, אנו יכולים ליצור אובייקט ) instance ) מהמחלקה WildKitten, ונשים console.log ב constructor , אנו נראה שקודם רץ ה constructor של המחלקה ממנו ירשנו ( super).
* כאשר יוצרים מחלקה יורשת, חייבים לקרוא למחלקת אב ( super )
* לא חייבים לקרוא ל super במתודות , כי יכול להיות שאנו רוצים לוגיקה שונה לגמריי. במידה ואנו כן קוראים, אז הלוגיקה של האב, מתווספת.



* ירושה ( multilevel inheritance )



ישנה גם ירושה מרובה ( multiple inheritance ) – שלא נתמכת ב TS



**חוק ההורשה:**

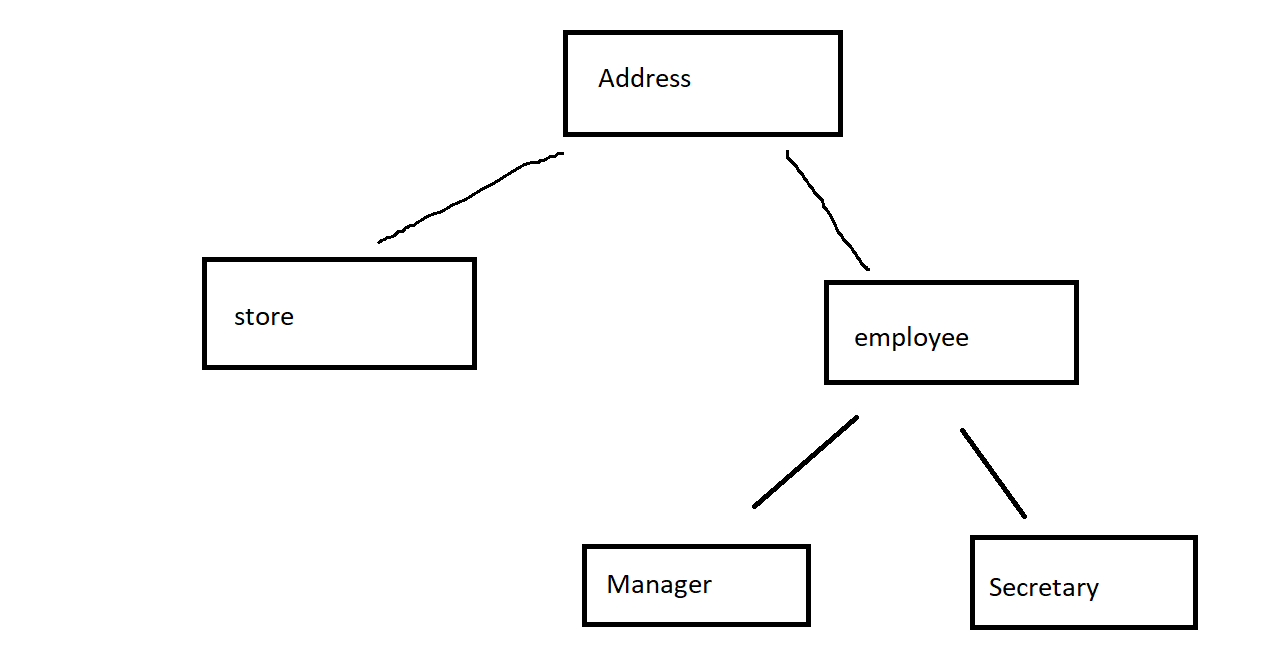
זה הוא חוק שקובע מתי ההורשה היא חוקית.

להלן החוק:

אם ניתן לומר במילים שהמחלקה היורשת **היא גם** – מחלקת הבסיס – ההורשה חוקית.

אם לא ניתן לומר זאת, או שזה לא נכון תמיד ב100% מהמקרים, אז ההורשה אינה חוקית.

דוגמה לעץ הורשה לא חוקי:



ה "point of view" שאנו אמורים לסגל – כאשר אנחנו יוצרים classים :

כאשר אנחנו יוצרים classים (מחלקות), **"הבן אדם שכותב את ה class, הוא לא הבן אדם שמשתמש ב class "** – זה אומר שבעצם אנחנו לא יוצרים אותם עבורנו, אלא עבור מתכנת אחר שאמור להשתמש בהם.

לדוגמה – אם אני כותב Motorcycle – זה עבור מתכנת אחר. מתכנת שידע ליצור אובייקט מהמחלקה הזו, מה הוא הולך לקבל, מה הם השדות ומה הן המתודות.

כך הוא יודע אם הוא יצר את השדות הדרושים (במידה והוא יורש), העביר את מה שצריך ל constructor, יצר מתודות רלוונטיות וכו.

**מחלקה אבסטרקטית**

זו מחלקה ממנה לא ניתן לייצר אובייקט, כי היא לא באמת מייצגת יישות אמיתית, אלא רק יישות אבסטרקטית , בכדי שנוכל לרשת אותה ולבנות יישויות אחרות.

ניתן ליצור מחלקה אבסטרקטית שיורשת ממחלקה אבסטרקיטת אחרת.

פונקציה אבסטרקטית :

חוק: אם אותה פונקציה (בעלת אותו שם ומשמעות) קיימת בכל היורשים, היא חייבת לראשונה להופיע בבסיס שלהם.

בעיה: רק אם מחלקת הבסיס הינה אבסטרקטית, יכול להיות (לא תמיד) שלא ניתן לתת תוכן לפונקציה, כי אנו לא יודעים לכתוב אותה במחלקה האבסטרקטית.

למשל (בהתייחס לפונקציה tune) – כיצד אנו יכולים לכתוב כיצד לעשות tuning , למנוע של כלי רכב כלשהו, בלי לדעת מה הוא כלי הרכב.

במקרה כזה, על הפונקציה להיות אבסטרקטית , זו היא פונקציה המכילה חתימה בלבד , ללא תוכן.

שני חוקים לגבי הפונקציה האבסטרקטית:

1. היא חייבת להיות במחלקה אבסטרקטית
2. מחלקה רגילה (לא אבסטרקטית) שיורשת ממחלקה אבסטרקטית, חייבת לממש (לדרוס ולכתוב תוכן משלה) את כל הפונקציות האבסטרקטיות שהיא קיבלה מהבסיס שלה.

Interface –ממשק

לא קשור לממש משתמש (user interface).

זו היא תבנית המכילה הצהרות בלבד על משתנים ו /או פונקציות.

זה מוגדר כ declaration . הוא מגדיר מה אמור להיות בתוך משהו.

מחלקה יכולה לממש interface , במקרה כזה היא תהיה חייבת להכיל את כל מה שיש בתוך ה interface ואפשר לכתוב Interface ולממש אותו בלי שום שקשר למחלקות.

שאלות ראיונות עבודה נפוצה :

מה ההבדל בין מחלקה אבסטרקטית לבין Interface

1. ייעוד :

מחלקה אבסטרקטית מיועדת להיות בסיס לעץ הורשה בלבד.

ממשק מיועד ויכול להכיל הצהרות על מה אמור להיות במחלקה או אובייקט.

1. נתונים טכניים:

ניתן לרשת ממחלקה אחת , אך ניתן לממש מספר ממשקים.

במחלקה אבסטרקטית יכולים להיות משתנים רגילים ופונקציות בעלות תוכן, בממשק יש רק הצהרות.