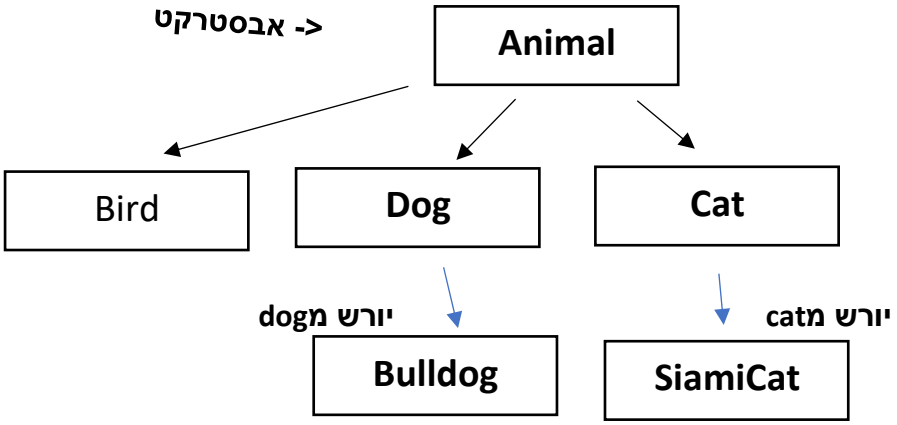
מבחן בנושא# C ו- בסיסי נתונים

חלק 1 - שאלות הגיון: (4 נקודות לשאלה. מקסימום 50 נקודות)

1)



כדאי להפוך את המחלקה Animal לabstract בגלל שבסופו של דבר כל המחלקות שיורשות מהמחלקה חייבות לממש את הפונקציות של ה'חיה' מה שבמקרה הזה הגיוני ומומלץ בהחלט כי יש לכל החיות מכנה משותף, אם יורשים ממחלקה אבסטרקטית אז חובה לממש את כל הפונקציות שיש בה.

2) **כן**, בדוטנט אפשר לרשת יותר ממחלקה אחת כי בסופו של דבר יכול להיות פונקציות בשתי מחלקות שונות שמתאימות לאותו סוג Class לדוגמה לPerson יכול להיות הרבה מאפיינים כמו Student,Teacher,Man,Girl

3) **כן**, בדוטנט ניתן לרשת יותר מInterface אחד אבל **למחלקה אחת(CLASS)**, אפשר לרשת הרבה interfaces לאותה מחלקה אבל לא ניתן יותר משתי מחלקות לרשת את אותם interfaces המחלקות יכולות לממש הרבה interfaces, את האינטרפייס חייב לממש לאותה המחלקה.

4) **לא ניתן** להצהיר <u>במחלקה לא אבסטרקטית</u> על <u>מתודה אבסטרקטית</u> מכיוון שאנו מחויבים להפוך את כל המחלקה היורשת למחלקה אבסטרקטית, ניתן לעשות מחלקה אבסטרקטית נפרדת לאותם מתודות ונהיה חייבים לממש אותם.

5) **לא ניתן** לכתוב <u>פונקציה לא אבסטרקטית</u> במחלקה אבסטרקטית מכיוון שברגע שכתבנו במחלקה abstract זה הופך את כל המחלקה אוטומטי לאבסטרקטית.

6) **ניתן** לכתוב פונקציה לא אבסטרקטית בInterface מכיוון שבinterface אנחנו מחויבים בסוג של "חוזה" לאותו מחלקה שאנחנו יורשים ממנה את הinterface, ופונקציה abstract היא מכילה בתוכה מידע בדומה לInterface

7) **לא ניתן** לעשות בInterface פונקציות שהם Public או בכללי כל מה שקשור לAccess Modifiers שזה Internal/Public/Protected/Private כי שוב, זה סוג של חוזה התחייבות מול המחלקה היורשת.

8) ההבדלים של הAccess Modifiers האלו:

Public
משתף את הפונקציות למחלקות וניתן לקרוא אותן באמצעות זה

Internal
ניתן לקרוא אך ורק בתוך הפרויקט

Private
ניתן לקרוא בתוך המחלקה

Protected
ניתן לקרוא בתוך המחלקה עצמה ולמחלקות היורשות ממנה

9) **לא חובה לממש** את הפונקציה של virtual בבנים היורשים,
פונקציות אבסטרקטיות **חובה לממש.**

10) אם כתבתי מתודה וירטואלית באבא ומימשתי בבן - האם זה override אם overload – **זה overload כיוון שאני לא דורסת שום פונקציה אחרת**.

אם כתבתי שתי פונקציות הזהות בשמותיהם אך שונות בפרמטרים האם זה override אם overload - **זה Override כיוון שאני דורסת את הפונקציה עם אותה השם ו"מחליפה אותה".**

אם כתבתי שתי פונקציות הזהות בשמותיהם ובפרמטרים אך שונות בטיפוס הערך המוחזר האם זה overload אם override – **זה Override כיוון שהשם של הפונקציה אותו הדבר והפונקציה האחרונה היא זאת שתקבע כי היא תדרוס את הקודמת.**

11) ניתן לעשות גם וגם .

12) בreadonly ניתן להכניס ערך בבנאי אך לאחר מכן לא ניתן לשנות אותו.

13) ההבדל בין מחלקה סטטית למחלקה לא סטטית זה שמי שיוכל לגשת למחלקה הסטטית זה רק מי שהוא סטטי – לצורך הדוגמה שדות סטטים ומתודות סטטיות יכולות לגשת למחלקה סטטית **אבל** מחלקה שהיא לא סטטית לא יכולה לגשת למחלקה סטטית.

חלק 2 - שאלות קודם קיים (15 נקודות לשאלה. מקסימום 30 נקודות)

בGIT. בקבצי CS

חלק 3 - שאלות הגיון בסיס נתונים: (4 נק' לשאלה. מקסימום 20 נק')

1) stored procedure זה בעצם מאגר בסיס הנתונים של הSQL, לנו בתור מפתחים/אנשי מקצוע ואנשים בעלי מאגרים גדולים של מידע זאת שיטה מעולה לאחסן ולשמור כמות ענקית של מידע, הSQL יודע לעשות הרצות של התוכנית ושיפורים ולעשות ביצועים יותר פשוטים כמו לראות את תוצאות הקוד בטבלאות, ברשומות יותר ברורות וכו', ניתן לקשר בין טבלאות ולהגיע לתוצאות מדויקות ותוך כדי לשמור על הסדר מה משויך למה, לדוגמה סועד שאומר למלצר שמעביר למטבח את ההזמנה.

2) כדאי להשתמש ב app.config לצורך אחסון הconnection string כיוון שזה יעזור לנו המון עם גישה לקובץ אחסנה, והרצה יותר פשוטה של התוכנית לפי השם קובץ כי הניתוב שכתבנו נמצא בapp.config, התקשרות ישירה לDatabase.

3) כמובן שכדי להשתמש בSQLITE, ההתקנה מאוד קצרה והתוכנה ממש פשוטה להעברה ולהבנה.

4) אפשר להשתמש ב **firebase** כמובן בתור בסיס נתונים פשוט ויש אפשרות לאחסן אותו בענן ואחרים יכולים לראות אותו.

5) מומלץ להשתמש ב**MSSQL** לצורך בסיס נתונים עשיר שתומך במס' שרתים עם אופציה של גיבוי ושכפול מכיוון שבMSSQL יש כל כך הרבה אופציות ואפשרויות לשחק עם הטבלאות ולקשר אותן אחת לשניה, בנוסף קיים בה אופציה לגיבוי ושכפול של הבסיס נתונים.

6) הבסיס נתונים שתומך בJSON בתצורתו הבסיסית הוא MSSQL

8) תשובה ג – Inner Join

9) תשובה א – Left Join

10) תשובה ד – Full Outer Join

11) תשובה ב – Right Join

12) תשובה ה – Cross Join

13) המטרה של מחלקת POCO בבסיס נתונים זה, כדאי לי לממש את Equals וGetHashCode על מנת שלא יהיו כפילויות בבסיס נתונים ולגבי האופרטור חשוב לעשות גם == וגם != על מנת להביא את שני האופציות שלא יהיה לנו מצב של שגיאה נצטרך להציג את שני המקרים .