**חלק 1 – שאלות הגיון:**

1.1-

יורשים למחלקת ‘Animal’ : ‘Dog’,’Cat’,’Bird’

יורשים למחלקת ‘Dog’ : ‘Bulldog’

יורשים למחלקת ‘Cat’ : ‘SiamiCat’

1.2-

כן, כי כנראה יש למחלקה הזו קוד משותף עם שאר המחלקות אם כי היישום והפעולות של השיטות שונה.

למשל ‘Animal’ מכילה שיטה בשם ‘MakeSound’ שבכל אחת משאר המחלקות מייושמת השיטה באופן טיפה שונה כי כל חיה משמיעה קול אחר.

2-

לא, נאמר שיש לנו מחלקה ‘A’ שיש לה יורשים ‘B’ ו’C’ , מחלקת ‘D’ **שיורשת** ממחלקת ‘B’ו’C’ קוראת לשיטה ‘Method’ שנמצאת במחלקת ‘A’. השיטה ‘Method’ מיושמת אחרת במחלקות ‘B’ ו’C’ ולכן לא ניתן להחליט בדיוק איזו שיטה תפעל במחלקה ‘D’.

3-

כן.

4-

לא.

5-

לא.

6-

כן.

7-

כן, internal אפשרי.

8-

public- כל אחד יכול לגשת למחלקה.

Private- רק המחלקה עצמה יכולה לגשת.

Protected- המחלקה והיורשות שלה יכולות לגשת.

Internal- המחלקה ואותו הפרויקט יכולים לגשת.

9-

לא.

כן.

10-

1.override

2.overload

3.overload

11-

גם וגם.

12-

Readonly

13-

לא ניתן לייצר מופע ממחלקה סטטית, ניתן לייצר מופע ממחלקה לא ססטית המכילה שיטות ושדות סטטיים.

14-

בנאי סטטי פועל בהתחלת התוכנית, בתוכו כותבים כל מה שלא קשור ישירות לשיטות או לשדות של התוכנית אבל יש בו צורך להפעלת התוכנית.

דוגמא: בדיבור מול שרת SQL ניתן לכתוב את החיבור לשרת בבנאי הססטי.

15-

Public int ID{get; set;}

16-

השימוש בpropery חוסך כתיבה של שיטות get וset עבור האובייקטים, הוא גם מאפשר לכותב לקבוע מהי רמת הגישה למשתנים.

17-

הוא מייצר backing field.

18-

Xml deserialization/serialization

19-

1.stack

2.queue

3.list

4.dictionary

20-

כי השימוש נשאר אותו דבר, רק האובייקטים עליהם הפעולה עובדת משתנים.

21-

Tostring- מדפיסה את האובייקט (שם המחלקה), דורסים כאשר רוצים להציג מידע אחר מהמחלקה.

Gettype- מחזירה את הסוג של המופע.

Equals- משווה את הסוג של האובייקטים, דורסים כאשר רוצים שוויון אחר.

Gethashcode- מחזיר את הhash code של המופע, דורסים כאשר רוצים להחזיר ערך אחר.

22-

כי הפונקציה gettype מחזירה את הסוג של האובייקט, אין הגיון בלדרוס אותה ולהחזיר ערך אחר.

23-

Icomparable.

24-

Icomparer.

25-

קטע הקוד בתוף הcatch יתבצע רק כאשר ייזרק האקספשיין הנכון.

Finally תמיד יבוצע.