1 - העמסה, overloading, זה כאשר אנחנו יוצרים מתודה עם אותו שם אבל פרמטרים שונים.
אנחנו יכולים להשתמש ב overloading כלפי בנאים כדי להגדיר בנאים שונים, כולם חייבים לקבל מספר פרמטרים שונה, כדי שיהיה לנו מספר דרכים לממש את אותו המופע.
2 - מקובל להשתמש ב overloading כדי ליצור מתודות שעושות פעולה זהה כלפי מבני נתונים שונים.
למשל מתודת sort שמסתרד array של int (ולכן גם תקבל array של int כפרמטר) ומתודת sort שתסדר array של strings (היא תקבל array של string). הרבה יותר הגיוני להשתמש בהעמסה מאשר ליצור מתודה חדשה הממשת את אותו ההיגיון.
3 - בעזרת המילה השמורה base - למשל

public ChildClass(var a, var b) : base (a)

4 - משמעות המילה השמורה virtual היא שפונקציה זאת תינתן לדריסה בעתיד - נשתמש בזה כאשר אנחנו רוצים להביא את האפשרות לדרוס את הפונקציה אבל לא מחייבים.
5 - אנחנו חייבים לממש מתודה שמוגדרת כ virtual
6 - לא חובה לממש מתודה מסוג virtual במחלקה היורשת ואם לא נעשה זאת נקבל את המימוש של מחלקת האם.
7 - לא מוסיפים virtual ל abstract כי הוא מחייב מימוש בכל מקרה, להבדיל מ virtual שרק מאפשר אותו.
8 - ההבדלים הם ש abstract לא מחייב מימוש במחלקה האבסטרקטית ומחייב מימוש במחלקה היורשת ו virtual מחייב מימוש במחלקת המקור ולא מחייב מימוש במחלקה היורשת.
9 - בגלל שנקרא לבנאי של מחלקת ה base תמיד לפני שנקרא לבנאי של המחלקה היורשת, ויש מספר בנאים במחלקת ה base אנחנו צריכים לציין לאיזה מן הבנאים אנחנו רוצים לקרוא - אחרת נקבל שגיא כי הקומפיילר לא מחליט בשבילנו.
10 - תמיד קודם יווצר מופע של מחלקת האם ורק לאחר מכן של המחלקה היורשת
11 - כל המחלקות יורשות מ Object
12 - במחלקת Object שכולם יורשים ממנה כיימת בין היתר הפונקציה ToString האיתרון של זה הוא שבגלל שזאת פונקציונליות כלכך שחיחה אנחנו תמיד נקבל מימוש דיפולטיבי גם בלי לדרוס אותה.
13 - כן, כל מחלקה יורשת מ Object וכל דבר הוא Object.
14 - בגלל שפרויקט קונסול הוא חסר ממשק משתמש (UI) הוא מיועד להרצה בקונסול, אנחנו מקבלים את המערך הארגומנטים כדי לשלוח ארגומנטים לפרויקט כשאנחנו קוראים לו בקונסולה ונעשה זאת על ידי כתיבת מיקום הפרויקט וקובץ הexe, רווח ואז את הארגומנטים הרלוונטיים.