Abstract
&
Interfaces

SYNTAX
שפה מייצרת חשיבה

אבסטקרט abstract

- abstractהיא מלה שמורה לפני שם של פונקציה או מחלקה.
- פונקציה אבסטרקטית היא הכרזה בלבד ללא מימוש.
- מחלקה המכילה פונקציה אבסטרקטית אחת היא מחלקה אבסטרקטית.
- במחלקה כזו מותר גם לכתוב פונקציות רגילות (עם מימוש).
- ממחלקה אבסטרקטית לא נוכל לייצר עצמים אבל נוכל לרשת ממנה.
- כל ירושה מחייבת לעשות override על כל הפונקציות האבסטרקטיות שלה או להיות אבסטרקטים.

SYNTAX
שפה מייצרת חשיבה

abstract example

```
abstract class Shape {
        public abstract void draw(); // אין מימוש
        public void print() {...} // פונקציה עם מימוש
 }

public class Square extends Shape { // יורשת  מאבסטרקטית
        public void draw() {....} // מחוייבים לממש
}
```

SYNTAX
שפה מייצרת חשיבה

יתרונות - abstract

- מחלקה מחלקה מופשטת כמו כלי רכב, כלי כתיבה, צורה, מרובע...
- אין "צורה" בעולם, אבל יש מלבן, משולש, מעגל ועוד.
- מחלקה מופשטת מגדירה תבנית ופעולות חובה. למשל לכל הצורות מגדירים פעולות חובה שיהיו בכל צורה כמו צייר, חשב שטח וכו' .
- הגדרת תבנית מחייבת לכל מי שירצה לרשת מאיתנו.
- נוכל לנצל את יתרונות הפולימורפיזם ולאפשר התייחסות כללית אחידה למרות שמחלקת האב כלל לא יישות בעולם.

דוגמה מעולם המעשה android

SYNTAX
שפה מייצרת חשיבה

```
abstract class BroadcastReceiver { //מחלקה כללית המקבלת שדר

        Public abtsract void onReceive(...);

}
```

```
class SmsReceiver extends BroadcastReceiver {

        public void onReceive(...) {

                //read the sms and do something with it

        }

}
```

ממשקים Interfaces

- ממשק הוא מבנה שכל הפונקציות שלו הן.**public abstract**
- את הממשק מיישמים באמצעות המילה השמורה**implements**
- בג׳אווה ניתן ליישם כמה ממשקים שנרצה.
- הממשק מגדיר התנהגות שניתן לאמץ.
- ניתן להגדיר אב מטיפוס הממשק אשר יצביע על עצם שממש אותו.
- כך נוכל לאגד מחלקות ללא אב משותף אבל עם התנהגות משותפת.

interface example

```
public interface Flyable { // הגדרת הממשק
        void fly(); //  public abstract אין צורך לציין
}

public class AirPlane extends Vehicle implements Flyable  {
    // other functions
    void fly() {.....} // מימוש הפונקציה המוגדרת בממשק היא חובה
}

public class Bird extend Animal implements Flyable {
        // other functions
        void fly {.....}
}
```

interface example - main

```
public static void main(String[] args)
{
        Flyable[] flyables ={new AirPlane(),new Bird(),new
Kite()}
        for(int i=0;i<flyables.length;i++) {
```

בעזרת הממשק ניתן לאגד ביחד מספר עצמים שלכאורה אין בינהם שום קשר מלבד ההתנהגות המשותפת.

אפשרות אחרת for - איטרטיבי

```
for (Flayble f : flyables) {
        f.fly()
}
```

Inner and anonymous class

- בג׳אווה ניתן להגדיר מחלקות פנימיות כלומר מחלקה בתוך מחלקה .
- למחלקה פנימית יש גישה לכל משתני ופונקציות המחלקה החיצונית שמכילה אותה ולכל המשתנים הלוקלים שהם.final
- ניתן גם להגדיר מחלקה פנימית אנונימית לצורך מימוש ממשק או מחלקה אבסטרקטית חד פעמי (אד-הוק) .
- ניתן לייצר מחלקה פנימית ללא שם (אנונימית) על ידי התחביר הבא:

Anonymous example

```
new שם המחלקה ממנה אני רוצה לרשת () {

        מימוש ו override של הפונקציות

}
```

אנו מקבלים ref מסוג
המחלקה שאותה ירשנו
לשימוש מיידי -
המחלקה שיצרנו כאן לא
נשמרה בשום מקום.

Anonymous interface implementation example

```
Flyable[] flyables = new Flyable[3];

flyables[0] = new AirPlane();

flyables[1] = new Bird()

flyables[2] = new Flyable() {

        public void fly() {System.out.println("I am flying my
new model airplane");

}
```

מימשנו את הממשק למחלקה חד פעמית
שכנראה ולא נרצה להגדיר ממנה יישויות
נוספות ולכן אין טעם לתת לה שם ולהגדיר
אותה בנפרד - פשטות

SYNTAX
שפה מייצרת חשיבה

Interfaces in the API

- Comparable ב API של ג׳אווה .
- הממשק Comparable מגדיר בר השוואה.
- (public int compareTo(Object other
- מיון ע״פ - natural othering מספר חיובי אם this גדול מ other, אם הם שווים 0, ומספר שלילי אם other גדול מ.this
- ניתן להשתמש בפונקציות שונות ב api כדוגמת Arrays.sort אשר יכולות למיין כל מערך בזכות ה״שפה האחידה״.

SYNTAX
שפה מייצרת חשיבה

Comparable example

```
public class Person implements Comparable {

        private String firstName;
        private String lastName;
        ... // פונקציות נוספות

        public int compareTo(Object other) {
                                        // String implements Comparable
                if(lastName.compareTo(other.lastName)>0) return 1;
                else if(lastName.compareTo(other.lastName)<0) return -1;
                else return firstName.compareTo(other.firstName);
    }
```

האנשים ימויינו לפי סדר אלפבתי של שמות המשפחה, אם אם זהים אז ימויינו לפי הסדר האלפבתי של השמות הפרטיים

SYNTAX
שפה מייצרת חשיבה

Android example

```
Interface OnClickListener {
        void onClick(View v);
}
```

```
class MyListener implements OnClickListener {
        Void onClick(View v) {
                //Do something with that click
}
```

```
Button b = new Button();
b.setOnClickListener( new OnClickListener() {
        void onClick(View v) {
                //do one time thing with the click
        }
});
```