

תוכנה 1

תרגול מס' 6: ממשקים
נעמה מאיר ומתי שמרת

1

ה Logger

- נפתח מערכת (פשוטה) לכתובת הודעות ל"קובץ לוג" במהלך ריצת התכנית
- ההודעות יהיו לצרכי בדיקה וגם כדי לעקוב אחרי פעילות המערכת
- על המערכת לתמוך בכתיבה למספר סוגי ערוצים (מסך, קובץ, בסיס נתונים, שרת קבצים ועוד)

2

ה Logger כמחלקה

```
public class Logger {  
    public void log(String msg) {  
        log(msg, 0);  
    }  
  
    public void log(String msg, int priority) {  
        System.out.println("Msg: "+ msg + ", Priority: " +  
            priority);  
    }  
}
```

- ממשש ע"י כתיבה למסך. ומה עם קובץ, שרת וכו'?

3

פתרון א'

- מחלקה אחת שתדע לטפל בכל האפשרויות
- יתרון:
 - הלקוח צריך להכיר רק מחלקה אחת (קוד פשוט)
- חסרונות:
 - קוד הלוגר מסורבל, נטייה לשגיאות
 - קשה להוסיף ערוצים נוספים
 - פגיעה במודולריות

פתרון ב'

- מחלקה עבור כל מימוש אפשרי
- יתרון:
 - מודולרי
- חסרון:
 - קוד הלקוח בלתי אפשרי

5

הלקוח בפתרון ב'

```
public class MyClass {  
  
    public void myMethod() {  
        ConsoleLogger cl = ... // get ConsoleLogger  
        if (cl != null) {  
            // use this logger  
        } else {  
            FileLogger fl = ... // get FileLogger  
            if (fl != null) {  
                // use this logger  
            } else {  
                ...  
                ...  
            }  
        }  
    }  
}
```

- ברור שזה אינו פתרון אפשרי
- נרצה להפריד בין הגדרת ה logger ובין המימושים השונים

6

הלקוח

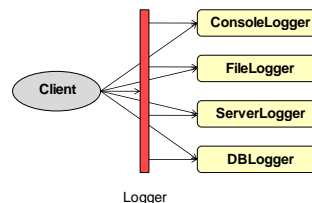
- כיצד נרצה שיראה קוד הלקוח?

```
public class MyClass {  
    public void myMethod() {  
        // obtain the logger  
        logger.log("Entering MyClass.myMethod()");  
        ...  
        logger.log("Exiting MyClass.myMethod()");  
    }  
}
```

- מה הטיפוס של logger?
■ ?ConsoleLogger, FileLogger, ServerLogger

7

באופן סכמטי



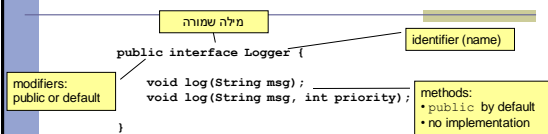
- הלקוח מכיר ממשק יחיד

8

ממשק interface

- הממשק עוזר לנו להפריד בין פונקציונאליות ובין המימוש שלה.
- מגדירים "חוזה" בין לקוח וספק ברמת השפה חתימות המתודות
- הלקוח לא צריך להכיר את הספק ובלבד שהוא מממש את הממשק
- הספק יודע אילו מתודות בדיוק עליו לממש ומהי חתימתן

הממשק Logger



- הגדרת ממשק דומה להגדרת מחלקה
- הנראות האפשרית היא public או default
- כל המתודות הן public (אין חובה לציין זאת)
- הממשק יכול הצהרה על מתודות ללא מימוש כלשהו

10

שימוש בממשק

- כעת נוכל לכתוב את הלקוח שלנו כפי שרצינו

```
public class MyClass {  
    public void myMethod() {  
        Logger logger = ...  
        logger.log("Entering MyClass.myMethod()");  
        ...  
        logger.log("Exiting MyClass.myMethod()");  
    }  
}
```

- הלקוח אינו תלוי במימוש ספציפי

מימוש הממשק

- עבור כל אחד מהמימושים (ערוצים) השונים נכתוב מחלקה המממשת כתיבה לערוץ זה
- מחלקות אלו יממשו את הממשק המוכר ע"י הלקוח יתרון:
- קוד לקוח פשוט - מכיר רק טיפוס אחד
- מודולריות במימוש
- קל להוסיף ערוצים חדשים

12

מימוש לדוגמא

```
FileLogger המחלקה ■  
ממשת את המנשק  
public class FileLogger implements Logger {  
    public FileLogger(String fileName) {...}  
    void log(String msg) {...}  
    void log(String msg, int priority) {  
        // write to the file  
    }  
    // more methods and variables  
}
```

חובה לספק מימוש לכל השירותים שבמנשק

ניתן להגדיר שירותים נוספים (למשל בואי)

מנשק הוא טיפוס

- בכל מקום בו השתמשנו בשם של מחלקה נוכל להשתמש בשם של מנשק
- הפניה למנשק תצביע לאובייקט של מחלקה כלשהי
- הממשת את המנשק
- דרך הפניה זו נוכל לקרוא רק לשירותים המוגדרים במנשק
- מחלקה יכולה לממש יותר ממנשק אחד

Polymorphism

```
public interface Inf1 {  
    void foo(Inf1 inf);  
}
```

נתון המנשק הבא:

ושתי מחלקות שמממשות אותו

```
public class Cls1 implements Inf1 {  
    public void foo(Inf1 inf) {  
        System.out.println("Cls1.foo(Inf1)");  
    }  
    public void foo(Cls1 cls1) {  
        System.out.println("Cls1.foo(Cls1)");  
    }  
}
```

```
public class Cls2 implements Inf1 {  
    public void foo(Inf1 inf) {  
        System.out.println("Cls2.foo(Inf1)");  
    }  
}
```

15

Polymorphism (המשך)

מה הפלט של הקוד הבא:

```
public class Main {  
    public static void main(String[] args) {  
        Cls1 cls1a = new Cls1();  
        Cls1 cls1b = new Cls1();  
        Cls2 cls2 = new Cls2();  
        Inf1 inf = cls1a;  
  
        cls1a.foo(cls1b);  
        inf.foo(cls1b);  
  
        cls1b.foo(inf);  
        cls1b.foo(cls1a);  
  
        cls2.foo(cls1a);  
        cls2.foo(cls2);  
    }  
}
```

16

ריבוי מנשקים

מחלקה יכולה לממש יותר ממנשק אחד

```
public class MyClass implements Inf1, Inf2, Inf3 {  
    ...  
}
```

- המחלקה חייבת לממש את כל השירותים בכל המנשקים
- נוכל להשתמש באובייקט של המחלקה בכל מקום בו נדרש אחד המנשקים או טיפוס המחלקה עצמה

ריבוי מנשקים

```
public class Main {  
    public static void main(String[] args) {  
        MyClass o = new MyClass();  
        Inf1 i1 = o;  
        Inf2 i2 = o;  
        Inf3 i3 = o;  
    }  
}
```

- בעזרת o נוכל לגשת לכל השירותים המוגדרים במחלקה
- דרך ההפניות האחרות נוכל לגשת רק לשירותים המתאימים למנשק

מתי נגדיר מנשק?

■ תמיד

■ לצורך תכנות נכון, תמיד נפריד בין הגדרה ומימוש

■ אף פעם

■ בשבוע הבא נראה שיש כלים אחרים ואולי מנשקים זה בכלל מיותר

✓ טוב נו, זה תלוי ...