

Hi-tech Company

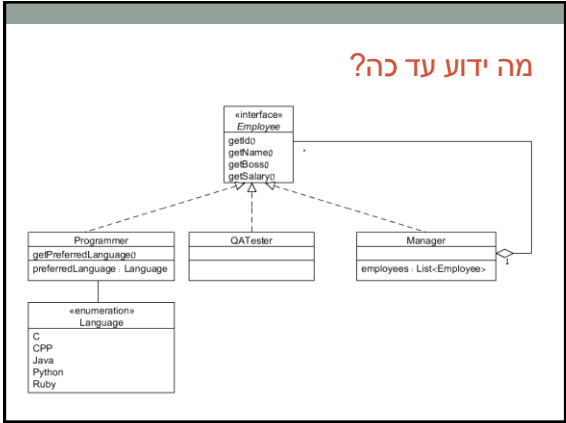
- בתרגיל זה נתרגל מספר נושאים אותם למדנו בשיעורים האחרונים:
- עיצוב ובניית מודל המורכב ממחלקות לתיאור סביבה מסוימת
- מנשקים, מחלקות מופשטות וירושה
- אוספים
- במסגרת התרגיל נכתוב תוכנית לחישוב שכר בחברת הייטק המורכבת ממספר סוגים של עובדים.

תוכנה 1

תרגול מספר 11:

תרגיל חזרה – חברת הייטק

בית הספר למדעי המחשב
אוניברסיטת תל אביב



עצבו מחלקות לייצוג עובדים בחברה על פי המפרט הבא:

- בחברת הייטק מצליחה ישנם 3 סוגי עובדים:
- תוכניתנים
- בודקי תוכנה
- מנהלים.
- לכל עובד יש שם, מזהה מספרי ובוס (מסוג מנהל).
- כל עובד מקבל משכורת.
- לכל מנהל יש רשימה של עובדים אותם הוא מנהל.
- לכל תוכניתן יש שפת תכנות מועדפת (מתוך רשימה אפשרית)

המשך המפרט:

- תוכניתנים ובודקי תוכנה מקבלים שכר בסיס אישי
- בודקי תוכנה מקבלים גם בונוס על כל באג שמצאו השבוע (בונוס אחיד לכל הבודקים פר באג).
- שכרו של כל מנהל נקבע כמספר העובדים שהוא מנהל ישירות * פקטור אישי.

Programmer	QATester	Manager
getId() getName() getBoss() getSalary() getPreferredLanguage()	getId() getName() getBoss() getSalary() getBugFound() getPerBugBonus()	getId() getName() getBoss() getSalary() getEmployeees()
wage : double preferredLanguage : Language name : String id : int boss : Manager	wage : double bugFound : int perBugBonus : double name : String id : int boss : Manager	employeeFactor : double employees : List<Employee> name : String id : int boss : Manager

המשך המפרט:

- תוכניתנים ובודקי תוכנה מקבלים שכר בסיס אישי
- בודקי תוכנה מקבלים גם בונוס על כל באג שמצאו השבוע (בונוס אחיד לכל הבודקים פר באג).
- שכרו של כל מנהל נקבע כמספר העובדים שהוא מנהל ישירות * פקטור אישי.

Programmer	QATester	Manager
getId() getName() getBoss() getSalary() getPreferredLanguage()	getId() getName() getBoss() getSalary() getBugFound() getPerBugBonus()	getId() getName() getBoss() getSalary() getEmployeees()
wage : double preferredLanguage : Language name : String id : int boss : Manager	wage : double bugFound : int perBugBonus : double name : String id : int boss : Manager	employeeFactor : double employees : List<Employee> name : String id : int boss : Manager

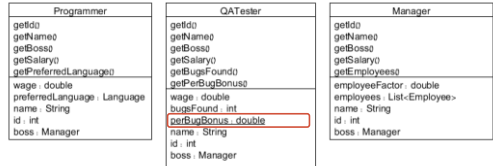
המשך המפרט:

- תוכניתנים ובודקי תוכנה מקבלים שכר בסיס אישי
- בודקי תוכנה מקבלים גם בonus על כל באג שמצאו השבוע (בנוס אחיד לכל הבודקים פר באג).
- שכרו של כל מנהל נקבע כמספר העובדים שהוא מנהל ישירות * פקטור אישי.

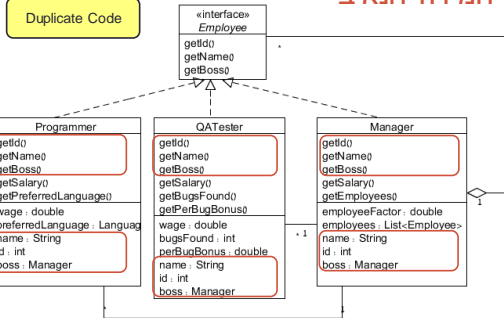


המשך המפרט:

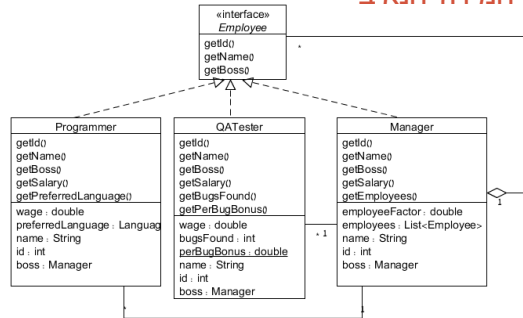
- תוכניתנים ובודקי תוכנה מקבלים שכר בסיס אישי
- בודקי תוכנה מקבלים גם בonus על כל באג שמצאו השבוע (בנוס אחיד לכל הבודקים פר באג).
- שכרו של כל מנהל נקבע כמספר העובדים שהוא מנהל ישירות * פקטור אישי.



המידול הנאיבי

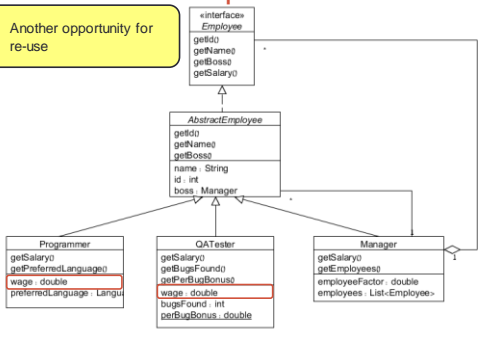


המידול הנאיבי



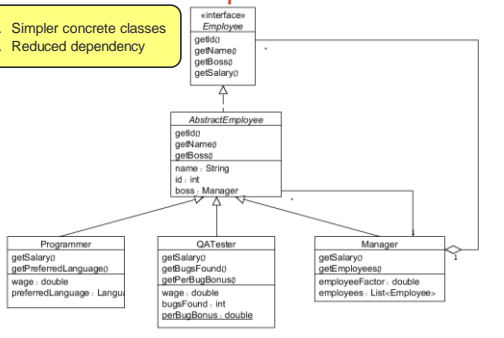
שלב 1 – עובד אבסטרקטי

1. Another opportunity for re-use



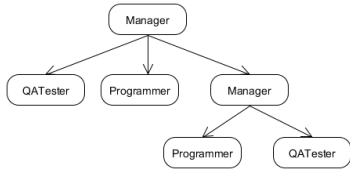
שלב 1 – עובד אבסטרקטי

1. Simpler concrete classes
2. Reduced dependency

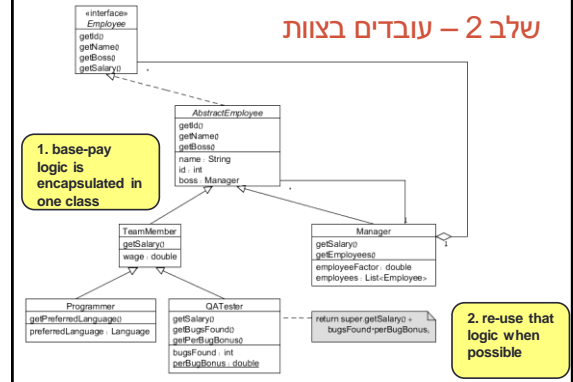


שלב 3 – plan ahead? (אופציונאלי)

- לפנינו מבנה היררכי (עץ)
- ייתכן שנרצה לעבור על המבנה בצורה אחידה
- מזכיר מאוד את תבנית העיצוב Composite (שינוי פשוט למחלקות)
- נשתמש בה (שינוי פשוט למחלקות)



שלב 2 – עובדים בצוות



1. base-pay logic is encapsulated in one class

2. re-use that logic when possible

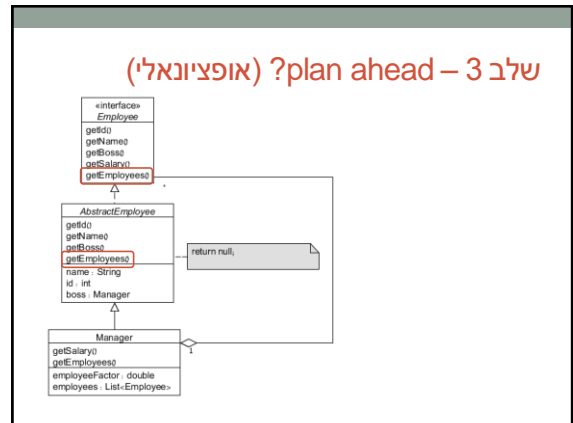
מה הלאה?

- לכתוב קוד!
- נעבור רק על החלקים המרכזיים
- שאר הקוד באתר



```
public interface Employee {
    public int getId();
    public String getName();
    public Manager getBoss();
    public double getSalary();
}
```

שלב 3 – plan ahead? (אופציונאלי)



Enumerated types

```
public enum language {
    C("C"),
    CPP("C++"),
    Java("Java"),
    Python("Python"),
    Ruby("Ruby");

    private final String displayName;

    private language(String name) {
        displayName = name;
    }

    @Override
    public String toString() {
        return displayName;
    }
}
```

עוד קוד

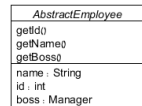
```
public abstract class AbstractEmployee implements Employee {
    private int id;
    private String name;
    private Manager boss;

    public AbstractEmployee(int id, String name, Manager boss) {
        this.id = id;
        this.name = name;
        this.boss = boss;
    }

    @Override
    public int getId() {
        return id;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public Manager getBoss() {
        return boss;
    }
}
```



פרטי מימוש...

- נרצה לוודא כי לעובד יש רק מנהל אחד.
- אין בעיה מצד העובד (משתנה יחיד למנהל)
- צריך לוודא שכאשר משנים מנהל מורידים את העובד מהרשימה המתאימה

```
public abstract class AbstractEmployee implements Employee {
    ...
    @Override
    public void setBoss(Manager newManager) {
        if(getBoss() != null)
            getBoss().removeEmployee(this);

        this.boss = newManager;
        if(getBoss() != null)
            getBoss().addEmployee(this);
    }
}
```

«interface» Employee
getId getName getBoss setBoss(newManager: Manager)

Enumerated types - usage

```
public class Programmer extends TeamMember {
    private Language preferredLanguage;

    public Programmer(int id, String name, Manager boss, double wage,
        Language preferredLanguage) {
        super(id, name, boss, wage);
        this.preferredLanguage = preferredLanguage;
    }

    public Language getPreferredLanguage() {
        return preferredLanguage;
    }
}
```

פרטי מימוש...

- תמיכה ב-Collections
- (ניתן ל-eclipse לעשות את העבודה.)
- טוב, נסתמך על שדה ה-id.

```
public abstract class AbstractEmployee implements Employee {
    ...
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        AbstractEmployee other = (AbstractEmployee) obj;
        if (id != other.id)
            return false;
        return true;
    }
}
```

פרטי מימוש...

- תמיכה ב-Hash
- (ניתן ל-eclipse לעשות את העבודה.)
- נסתמך על שדה ה-id.

```
public abstract class AbstractEmployee implements Employee {
    ...
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + id;
        return result;
    }
}
```

חישובי שכר

- חישוב שכר עפ"י שכר בסיס

```
public class TeamMember extends AbstractEmployee {
    private double wage;

    public TeamMember(int id, String name, Manager boss,
        double wage) {
        super(id, name, boss);
        this.wage = wage;
    }

    @Override
    public double getSalary() {
        return wage;
    }
}
```

חישובי שכר

- למנהל חישוב שכר ייחודי

```
public class Manager extends AbstractEmployee {
    @Override
    public double getSalary() {
        return employeeFactor * employees.size();
    }
}
```

עוד דרישות:



- כתבו תוכנית המייצרת אובייקטים של עובדים עם נתונים אקראיים ושומרת אותם בשלוש רמות היררכיות לפי הפירוט הבא:
 - בראש ההיררכיה נמצא המנכ"ל שהינו מנהל
 - מתחתיו בהיררכיה יש 5 מנהלים
 - מתחת לכל מנהל מצויים בהיררכיה 10 תוכניתנים או בודקי תוכנה (בהסתברות שווה).
- לאחר מכן, התוכנית תדפיס את פרטי 3 העובדים עם המשכורת הגבוהה ביותר בכל רמה היררכית.

חישובי שכר

• חישוב שכר עפ"י שכר בסיס + בונוס

```
public class QATester extends TeamMember {
    private static double PER_BUG_BONUS = 100.0;
    private int bugsFound = 0;

    public QATester(int id, String name, Manager boss, double wage) {
        super(id, name, boss, wage);
    }

    public void incrementBugs() {...}
    public int getBugsFound() {...}

    @Override
    public double getSalary() {
        return super.getSalary() + getBugsFound() * PER_BUG_BONUS;
    }
}
```

איך מייצרים דו"ח?

• שימוש ב- instanceof במתודת יצירת דו"ח

• שימוש ב- toString (או מתודה ייעודית)
• תלוי במספר מצומצם של פורמטים/דו"חות?

• שימוש במחלקה ייעודית לכל דו"ח
• תלוי בכך שאין שינויים רבים במחלקות

דוגמא לפלט:

```
CEO:
ID: 1          Name: Osnat Nachshoni  Boss: None          Salary: 49740.43  Employees: 5
Managers:
ID: 13         Name: Dana Levi          Boss: Osnat Nachshoni  Salary: 30395.94  Employees: 10
ID: 24         Name: Neta Aharoni       Boss: Osnat Nachshoni  Salary: 29222.68  Employees: 10
ID: 35         Name: Dana Levi          Boss: Osnat Nachshoni  Salary: 25677.13  Employees: 10
Team members:
ID: 32         Name: Dana Nachshoni     Boss: Neta Aharoni    Salary: 20675.38  Language: Java
ID: 40         Name: Dana Nachshoni     Boss: Dana Levi       Salary: 19595.35  Language: C++
ID: 16         Name: Tom Cohen          Boss: Dana Levi       Salary: 19509.67  Language: Ruby
```

toString()

```
public class QATester extends TeamMember {
    ...
    @Override
    public String toString() {
        return super.toString() + "\tBugs found: " + getBugsFound();
    }
}
```

toString()

```
public abstract class AbstractEmployee implements Employee {
    ...
    @Override
    public String toString() {
        StringBuilder str = new StringBuilder();
        str.append("\tID: ").append(id);
        str.append("\tName: ").append(name);
        str.append("\tBoss: ");
        if (getBoss() != null)
            str.append(getBoss().getName());
        else
            str.append("None");
        str.append("\tSalary: ");
        str.append(String.format("%.2f", getSalary()));

        return str.toString();
    }
}
```

toString()

```

public abstract class AbstractEmployee implements Employee {
    ...
    @Override
    public String toString() {
        StringBuilder str = new StringBuilder();
        str.append("ID: ").append(id);
        str.append("\tName: ").append(name);
        str.append("\tBoss: ");
        if (getBoss() != null)
            str.append(getBoss().getName());
        else
            str.append("None");
        str.append("\tSalary: ");
        str.append(String.format("%.2f", getSalary()));
        return str.toString();
    }
}

```

עוד דרישות:



- כתבו תוכנית המייצרת אובייקטים של עובדים עם נתונים אקראיים ושומרת אותם בשלוש רמות היררכיות לפי הפירוט הבא:
 - בראש ההיררכיה נמצא המנכ"ל שהינו מנהל מתחתיו בהיררכיה יש 5 מנהלים
 - מתחת לכל מנהל מצויים בהיררכיה 10 תוכניתנים או בודקי תוכנה (בהסתברות שווה).

• לאחר מכן, התוכנית תדפיס את פרטי 3 העובדים עם המשכורת הגבוהה ביותר בכל רמה היררכית.

THE END

הקוד נמצא במלואו באתר הקורס

Sorting by salary

• נגדיר השוואה מתאימה:

```

public class SalaryComparator implements Comparator<Employee> {
    @Override
    public int compare(Employee o1, Employee o2) {
        return Double.compare(o2.getSalary(), o1.getSalary());
    }
}

```

Reverse sort

• כעת נוכל לייצר את הדו"ח

```

public static void printTopPaid(List<Employee> employees) {
    Collections.sort(employees, new SalaryComparator());
    for(int i=0; i<3; ++i)
        System.out.println(employees.get(i));
}

```