תוכנה 1 בשפת Java
שיעור 6: Class design example

מיכל קליינבורט

בית הספר למדעי המחשב

אוניברסיטת תל אביב

הדרישות

- עלינו לכתוב מערכת תוכנה שמייצגת חברה
  - 2 סוגי עובדים: עובד רגיל (standard employee), עובד זמני (temp)
    - סוגי עובד רגיל: אנשי מכירות (salesperson), אנשי כספים (finance worker) , מנהל (regional manager)

  לכל עובד נשמור: שם, טלפון, ת.ז. ועוד פרטים נוספים
  - ישנו צוות אחר שמפתח את הקוד של מערכת חישוב השכר: כדי לחשב שכר לעובד יש להבין עבור כל עובד מה שכר הבסיס שלו (computeBaseSalary), מה סך העמלות שקיבל בחודש (computeCommission)
  - כדי לאפשר פיתוח מקביל של התוכנה שלנו ושל התוכנה עבור מערכת חישוב השכר נסכם על מנשק בשם ISalary.
  - ישנה גם מערכת שמגדירה הרשאות לכל עובד:
    - יש צורך במנשק ISecurityLevel, שלפיו יקבעו ההרשאות עבור העובד

Credit: Mashable Composite: Getty Images/NBCUniversal

מה יכיל המנשק ISalary ?

```
public interface ISalary {
      public abstract int computeBaseSalary();
      public abstract int computeCommissions();
}
```

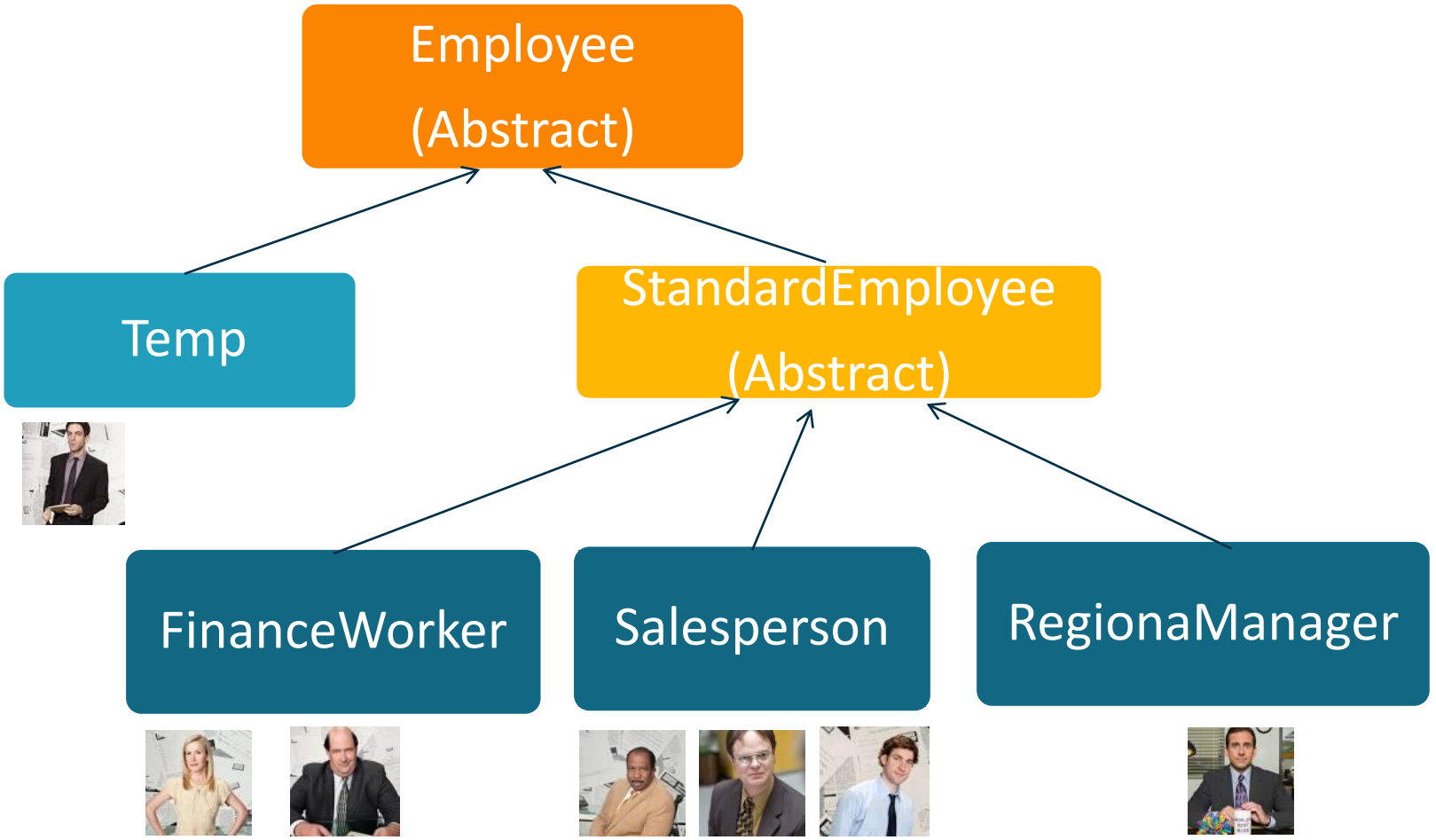
המחלקה Employee

```
Public abstract class Employee implements ISalary, ISecurityLevel
{
      private String name;
      private String phonenum;
      private String id;
      //many other fields

      //getters
      //setters
}
```

בחרנו להגדיר את Employee כמחלקה אבסטרקטית

המחלקות היורשות מ Employee



המחלקה Employee

בגלל ש Employee מממשת את המנשק ISalary, על כל מחלקה שאינה אבסטרקטית בהירכיה של Employee לממש את המתודות:

```
public abstract int computeBaseSalary();
public abstract int computeCommissions();
```

שמוגדרות במנשק.

כנ"ל לגבי המתודות המוגדרות במנשקים האחרים שEmployee תממש.

מדוע לא להגדיר פשוט את המתודות הנ"ל כחלק מהמחלקה האסטרקטית Employee?

יש הגיון בהפרדה שנוצרת בעזרת המנשק: מבחינת תוכנת חישוב השכר כל המידע הרב השמור באובייקט מטיפוס שיורש מ Employee אינו רלוונטי. כל מה שחשוב מבחינתם הוא הפונקציונליות של חישוב השכר שעלולה היות מורכבת ולהשתנות בין סוגי עובדים שונים.