

הוראת ירושה

מדריך למורה

רקע

לדעתי יש להבדיל בין שני מקרים עיקריים לשימוש במנגנון הירושה. במקרה ראשון קיימת מחלקה שמממשת עצם כלשהו. אנו זקוקים להוסיף למימוש זה דברים נוספים או לשנת מספר דברים. במקרה כזה כדאי לכתוב מחלקה שיורשת מהמחלקה הקיימת ובה להכניס את השינויים הדרושים. במקרה השני יש צורך לבנות מערכת המורכבת ממספר רב של עצמים מסוגים שונים. לעצמים בין היתר קיים מספר רב של תכונות ופעולות דומות או זהות. על מנת להימנע מקוד חוזר כדאי לבנות היררכיה של מחלקות על בהם ימומשו את החלקים המשותפים לעצמים.

לדעתי יותר פשוט להסביר את הרציונל של הירושה בעזרת המקרה הראשון. לזה מוקדשים דפי העבודה ExTurtle ו-ExTurtle2.

הידע הנדרש

כדאי ללמד את הנושא הזה לאחר הוראת יסודות מדעי המחשב: התלמידים מכירים אלגוריתמיקה בסיסית וכתיבת מחלקות. כמו כן ההנחה היא שהתלמידים מכירים את המחלקה Turtle מהספרייה unit4. במידה והם לא מכירים, כדאי להקדיש שיעור של 45 דקות לתרגול השימוש במחלקה Turtle.

מפגש ראשון (90 דקות)

במפגש הראשון יש לתאר את הדרישה: במחלקה Turtle הקיימת בספרייה unit4 חסרה אפשרות למדוד מרחק שהצב עבר ממקום כלשהו. רוצים להוסיף את אפשרות הזאת. כמו כן רוצים לשמר את שאר הפעולות של הצב ללא שינוי. נתון API של המחלקה החדשה MyTurtle ובו הפעולות העיקריות של המחלקה Turtle ושתי הפעולות החדשות reset ו-getDistance שאמורות לתת מענה לצורך מדידת המרחק שהצב עבר. לפני שהתלמידים ייגשו לכתובת תוכנה מומלץ לנהל דיון איך לממש מחלקה MyTurtle. אין צורך להסביר ירושה. אפשר לממש את המחלקה בעזרת הגדרת תכונה מסוג Turtle ושימוש בפעולות של המחלקה הזאת.

```
public class MyTurtle
{
    private double distance;
    private Turtle t;
    public MyTurtle()
    {
        t = new Turtle();
        this.distance = 0;
    }
    public void moveForward(double x)
    {
        this.distance += x;
    }
}
```

```

        t.moveForward(x);
    }
    public void moveBackward(double x)
    {
        this.distance += x;
        t.moveForward(-x);
    }
    public double getDistance()
    {
        return this.distance;
    }
    public void reset()
    {
        distance = 0;
    }
    public void turnLeft(double x)
    {
        t.turnLeft(x);
    }
    public void turnRight(double x)
    {
        t.turnRight(x);
    }
    public void tailDown()
    {
        t.tailDown();
    }
    public void tailUp()
    {
        t.tailUp();
    }
    public void setDelay(int x)
    {
        t.setDelay(x);
    }
}

```

לאחר דיון בכיתה התלמידים מתבקשים לממש את המחלקה MyTurtle ולכתוב תוכנת בדיקה. פעולה שבודקת מימוש המחלקה MyTurtle אמורה להפעיל את כל הפעולות שנמצאות בממשק כולל הפעולות המקוריות והפעולות החדשות.

כאשר המחלקה MyTurtle נכתבה ונבדקה אפשר לגשת למימוש המשימה השנייה.

במשימה השנייה התלמידים מתבקשים לממש מרוץ צבים - משימה יחסית מורכבת מבחינה אלגוריתמית. חשוב להקדיש מקום משמעותי לדיון על אלגוריתם לפתרון הבעיה. מכיוון שהפעילות הזאת מוקדשת לתמי"ע, לא נתייחס כאן להתמודדות עם הצד האלגוריתמי.

בדיון חשוב לציין בפני תלמידים יתרון של המחלקה MyTurtle בהשוואה למחלקה Turtle: עצם שמייצג כל אחד מן הצבים מכיל את המרחק שהצב עבר מקו הזינוק. אילו העצמים היו מסוג Turtle במימוש מרוץ צבים היה צורך במבנה נתונים נוסף, למשל במערך לשמירת מרחקים שכל אחד מהצבים עבר.

למימוש התרגיל לא יהיה מספיק זמן במפגש ראשון. מומלץ להשאיר אותו לשיעורי בית או להקדיש שיעור נוסף.

מפגש שני (90 דקות)

בתחילת המפגש מומלץ לחזור על המימוש של המחלקה MyTurtle : מה היו מטרתיו (הוספת מד מרחק), יתרונות של שימוש חוזר בקוד (לא היה צורך לממש צב מחדש) וחסרונות המימוש באמצעות הכלה. החיסרון הבולט הוא שיש צורך לשכתב במחלקה MyTurtle מספר פעולות של המחלקה המקורית כגון turnLeft, tailDown, setDelay ללא צורך ממשי לשנות או להוסיף משהו בהם. הפעולות האלה פשוט מזמנות את הפעולות התואמות ממחלקה Turtle.

כאן המקום להסביר את מנגנון הירושה. לא נתייחס כאן להסבר. אציין רק שביצוע דף העבודה ExTurtle2 דורש הסבר לדברים הבאים :

- הגדרת מחלקה שיורשת ממחלקה אחרת
- הגדרת פעולה מחדש (דריסה, override)
- שימוש ב-super() לזימון פעולה בונה ממחלקת על
- שימוש בהפניה super לזימון פעולה ממחלקת על

תוך כדי ביצוע דף עבודה חשוב לא לעבוד על הפרויקט שנשמר מדף העבודה הקודם, אלא ליצור העתק של הפרויקט. כך אפשר להריץ פרויקט ראשון והשני ולהשוות את התוצאות. בפרויקט השני ניתן לעשות שינויים אך ורק במחלקה MyTurtle. המחלקה לבדיקה והמחלקה שמממשת את מרוץ הצבים אמורות להישאר ללא שינוי.

```
public class MyTurtle extends Turtle
{
    private double distance;
    public MyTurtle()
    {
        super();
        this.distance = 0;
    }
    @Override
    public void moveForward(double x)
    {
        this.distance += x;
        super.moveForward(x);
    }
    @Override
    public void moveBackward(double x)
    {
        this.distance += x;
        super.moveForward(-x);
    }
}
```

```
}  
public double getDistance()  
{  
    return this.distance;  
}  
public void reset()  
{  
    distance = 0;  
}  
}
```

שימוש ב-super() בפעולה בונה לא הכרחי. אבל מומלץ לשמור לאחר כך את הדיון בסוגיית פעולה בונה ברירת מחדל ולכן כדאי להשאיר זימון מפורט לפעולה הבונה ממחלקת על. חשוב שתלמידים יריצו את בשני הפרויקטים גם את תוכנת הבדיקה וגם את מרוץ הצבים ויבדקו האם בשני המקרים מגיעים לאותן התוצאות. בסיכום ניתן לנהל דיון על יתרונות המימוש בעזרת הירושה: קוד חדש לא מכיל שכתובים מיותרים, אלא רק את הפעולות החדשות ואת אלא שיש צורך לשכתב אותם.