תרגול מספר 8: הורשה

ברישת בית הספר למדעי המחשב אוניברסיטת תל אביב

Iranian software development is not something we've heard much about.

The class extends an existing class:

```java
public class DrumTurtle extends Turtle {
    // A drum turtle is a turtle, a drum turtle can't move
    // forward in a straight line but instead it moves
    // in a zig zag fashion.
    public DrumTurtle() {
        super();
    }

    @Override
    public void moveForward(double steps) {
        super.moveForward(steps);
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class SmartTurtle extends Turtle {
    // A logo turtle that knows how to draw squares
    public SmartTurtle() {
        super();
    }

    public void drawSquare(int edge) {
        for (int i = 0; i < edge; i++) {
            moveForward(edge);
            turnLeft(90);
        }
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class Drisservice extends Service {
    // A service class that can be used as a parent class for
    // other service classes
    public Drisservice() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class ProtectedTurtle extends Turtle {
    // A protected turtle that can be used as a base class for
    // other turtle classes
    public ProtectedTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class PrivateTurtle extends Turtle {
    // A private turtle that can be used as a base class for
    // other turtle classes
    public PrivateTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class ProtectedTurtle extends Turtle {
    // A protected turtle that can be used as a base class for
    // other turtle classes
    public ProtectedTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class PrivateTurtle extends Turtle {
    // A private turtle that can be used as a base class for
    // other turtle classes
    public PrivateTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class ProtectedTurtle extends Turtle {
    // A protected turtle that can be used as a base class for
    // other turtle classes
    public ProtectedTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class PrivateTurtle extends Turtle {
    // A private turtle that can be used as a base class for
    // other turtle classes
    public PrivateTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class ProtectedTurtle extends Turtle {
    // A protected turtle that can be used as a base class for
    // other turtle classes
    public ProtectedTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class PrivateTurtle extends Turtle {
    // A private turtle that can be used as a base class for
    // other turtle classes
    public PrivateTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class ProtectedTurtle extends Turtle {
    // A protected turtle that can be used as a base class for
    // other turtle classes
    public ProtectedTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class PrivateTurtle extends Turtle {
    // A private turtle that can be used as a base class for
    // other turtle classes
    public PrivateTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class ProtectedTurtle extends Turtle {
    // A protected turtle that can be used as a base class for
    // other turtle classes
    public ProtectedTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.

The class extends the existing class:

```java
public class PrivateTurtle extends Turtle {
    // A private turtle that can be used as a base class for
    // other turtle classes
    public PrivateTurtle() {
        super();
    }
}
```

Can you think of any other ways to use the Turtle class to create artistic or educational content? If so, please share any ideas you have.
In the client side:

- We presented the IPoint interface and showed 3 different implementations for it.
- We showed that clients dependent on the IPoint interface only, and do not know about the implementations of these classes will be oblivious to future changes in the provider code.
- Using interfaces saves a duplicated code in the client side, as the same code works correctly with different providers (polymorphism).

In the provider side:

- In contrast, the delegation mechanism saves a duplicated code in the provider side.
- "IPoint interface takes a class instead of returning it.
  Both providers share the same code.
- Trying to identify the duplicated code between 3 implementers of the IPoint interface and concentrate these pieces in a base class that will be common for the three implementations.

Abstract classes:

- An abstract class is a class that contains methods that must be implemented by its subclasses.
  - Abstract methods cannot be called directly in an instance of the abstract class.
  - The abstract class must be extended by concrete classes.
  - When creating an object of an abstract class, you must create an object of its subclass.

Example of an abstract class:

```java
public abstract class A {
    public abstract void f();

    public void g() {
        System.out.println("A.g!!");
    }
}

public class B extends A {
    public void g() {
        System.out.println("B.g!!");
    }
}
```

```java
A a = new A();
A a = new B();
```
<table>
<thead>
<tr>
<th>CartesianPoint</th>
<th>PolarPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public void rotate(double angle)</code></td>
<td><code>public void rotate(double angle)</code></td>
</tr>
<tr>
<td><code>double currentTheta = Math.atan2(y,x);</code></td>
<td><code>double currentTheta = Math.atan2(y,x);</code></td>
</tr>
<tr>
<td><code>x = currentRho * Math.cos(currentTheta+angle);</code></td>
<td><code>dx = currentRho * Math.cos(currentTheta+angle);</code></td>
</tr>
<tr>
<td><code>y = currentRho * Math.sin(currentTheta+angle);</code></td>
<td><code>dy = currentRho * Math.sin(currentTheta+angle);</code></td>
</tr>
<tr>
<td><code>y += dy;</code></td>
<td><code>y += dy;</code></td>
</tr>
<tr>
<td><code>public void translate(double dx, double dy)</code></td>
<td><code>public void translate(double dx, double dy)</code></td>
</tr>
<tr>
<td><code>x += dx;</code></td>
<td><code>x += dx;</code></td>
</tr>
</tbody>
</table>

**Extract Superclass Refactoring**

-可用于自动执行此过程的工具是名为“提取超类”（Refactoring）的工具，它可以在Eclipse中使用。
-该工具尚未完全实现。

-“extract superclass”功能。
-此功能用于从现有类中提取新的超类，以减少代码重复并提高可维护性。

-在Eclipse中使用“extract superclass”功能时，应注意以下几点：
  - 确保代码结构清晰，以便在提取超类后进行重构。
  - 使用工具时应谨慎，避免引入错误或影响应用程序的正常运行。
  - 在重构后，需要对新超类及其子类进行测试，以确保功能正常。

-在实际开发过程中，应该根据具体需求和项目规模灵活使用“extract superclass”功能，以提高代码质量和开发效率。
אתחולים ובנאים

- יצירת מופע חדש של עצם כוללת:
  - הקצאת זיכרון
  - אתחול
  - פעלת בנאים והשמה לשדות

במסגרת ריצת הבנאי נקראים גם הבנאי/ים של מחלקת הבסיס

תהליך זה מבלבל כי לשדה מסוים ניתן לבצע השמות גם "י אתחול" ו"י מספר בנאים (אחרון קובע)

בשקפים הבאיםaremos במדויק את התהליך

נעזר בדוגמה מה הסדר ביצירת מופע של מחלקה?

1. שלב ראשון: הקצאת זיכרון לשדות העצם והצבת ערכי ברירת מחדל
2. שלב שני: נקרא הבנאי (לפי חתימת new) והאלגוריתם הבא מופעל:
   1._bind constructor parameters.
   2. If explicit this(), goto Step 5.
   3. Call recursively the implicit or explicit super(...) [except for Object because Object has no parent class]
   4. Execute the explicit instance variable initializers.
   5. Execute the body of the current constructor.

דוגמה

```java
public class Employee extends Object {
    private String name;
    private double salary = 15000.00;
    private Date birthDate;

    public Employee(String n, Date DoB) {
        // implicit super();
        name = n;
        birthDate = DoB;
    }

    public Employee(String n) {
        this(n, null);
    }
}
```

```java
public class Manager extends Employee {
    private String department;

    public Manager(String n, String d) {
        super(n);
        department = d;
    }
}
```

הרצת הדוגמה

מהקרה כהרי הJVM מיראי את השורה

Manager m = new Manager("Joe Smith", "Sales");

שלב ראשון: הקצאת זיכרון לשדות העצם והצבת ערכי ברירת מחדל

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Bind parameters.</td>
</tr>
<tr>
<td>2.</td>
<td>If explicit this(), goto Step 5.</td>
</tr>
<tr>
<td>3.</td>
<td>super().</td>
</tr>
<tr>
<td>4.</td>
<td>explicit var. init.</td>
</tr>
<tr>
<td>5.</td>
<td>Execute body.</td>
</tr>
</tbody>
</table>

enedor

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Bind parameters.</td>
</tr>
<tr>
<td>2.</td>
<td>If explicit this(), goto Step 5.</td>
</tr>
<tr>
<td>3.</td>
<td>super().</td>
</tr>
<tr>
<td>4.</td>
<td>explicit var. init.</td>
</tr>
<tr>
<td>5.</td>
<td>Execute body.</td>
</tr>
</tbody>
</table>

 algumas ת اللجنة הסדר ביצירת מופע של מחלקה?

1. שלב ראשון: הקצאת זיכרון לשדות העצם והצבת ערכי ברירת מחדל
2. שלב שני: נקרא הבנאי (לפי חתימת new) והאלגוריתם הבא מופעל:
   1._bind constructor parameters.
   2. If explicit this(), goto Step 5.
   3. Call recursively the implicit or explicit super(...) [except for Object because Object has no parent class]
   4. Execute the explicit instance variable initializers.
   5. Execute the body of the current constructor.

דוגמה

```java
public class Employee {
    private String name;
    private double salary = 15000.00;
    private Date birthDate;

    public Employee(String n, Date DoB) {
        // implicit super();
        name = n;
        birthDate = DoB;
    }

    public Employee(String n) {
        this(n, null);
    }
}
```

```java
public class Manager extends Employee {
    private String department;

    public Manager(String n, String d) {
        super(n);
        department = d;
    }
}
```