

תוכנה 1 - תרגול 12

סיכום

בחינה באופק!

■ הבחינה תכלול את כל הנושאים שכיסינו במהלך

הסמסטר:

■ כל ההרצאות

■ כל תרגולים

■ כל תרגילי בית

■ חומר סגור

■ חלק פתוח

■ חלק אמריקאי

קצת על מנשקים

- מנשק יכול להרחיב יותר ממנשק אחד
- שירותים במנשק הם תמיד ציבוריים, וכברירת מחדל מופשטים

```
public interface MyInterface {  
  
    public abstract int foo1(int i);  
  
    int foo2(int i);  
}
```

The modifiers of foo1 and foo2 are the same.

מנשקים

```
public interface Foo {  
    public void bar() throws Exception;  
}
```

```
public class FooImpl implements Foo {  
    public void bar() {  
        System.out.println("No exception is thrown");  
    }  
}
```

```
public static void main(String args[]) {  
    Foo foo = new FooImpl();  
    foo.bar();  
}
```

האם הקוד מתקמפל? אם לא, למה?
אם כן, האם יש שגיאת ריצה? אם יש, למה?
אחרת, מה הפלט?

שגיאת קומפילציה:
"Unhandled exception type Exception"

מנשקים - המשך

```
public interface Foo {  
    public void bar() throws Exception;  
}
```

```
public class FooImpl implements Foo {  
    public void bar() {  
        System.out.println("No exception is thrown");  
    }  
}
```

```
public static void main(String args[]) {  
    FooImpl foo = new FooImpl();  
    foo.bar();  
}
```

האם הקוד מתקמפל? אם לא, למה?
אם כן, האם יש שגיאת ריצה? אם יש, למה?
אחרת, מה הפלט?

פלט:

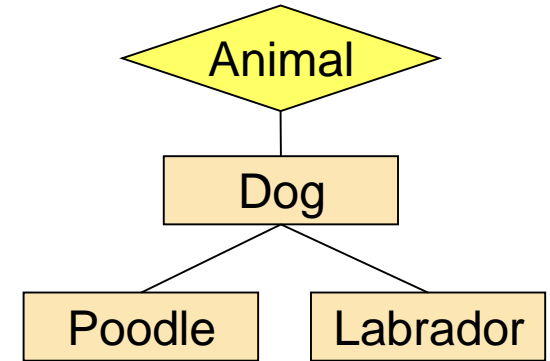
"No exception is thrown"



מנשקים וירושה

Consider the following class hierarchy:

```
Interface Animal {...}
class Dog implements Animal {...}
class Poodle extends Dog {...}
class Labrador extends Dog {...}
```



Which of the following lines (if any) will not compile?

- Poodle poodle = new Poodle();
- Animal animal = (Animal) poodle;
- Dog dog = new Labrador();
- animal = dog;
- poodle = dog;

- **Compilation Error**

Type mismatch: cannot convert from Dog to Poodle

poodle = (Poodle) dog;

-No compilation error

-Runtime Exception

Labrador labrador = (Labrador) dog;

-No compilation error

-No Runtime Exception



מנשקים וירוושה

```
class A {  
    public void print() {  
        System.out.println("A");  
    }  
}
```

האם יש שגיאה?

```
class B extends A implements C {  
}
```

```
interface C {  
    void print();  
}
```

אין שגיאות קומפילציה

public כברירת מחדל



מנשקים וירושה

```
class A {  
    void print() {  
        System.out.println("A");  
    }  
}
```

האם יש שגיאה?

```
class B extends A implements C {  
}
```

```
interface C {  
    void print();  
}
```

שגיאת קומפילציה:
The inherited package method A.print() cannot hide the public abstract method in C

The following table shows the access to members permitted by each modifier

Modifier	Access Level			
	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
No modifier	Y	Y	N	N
private	Y	N	N	N

A class may be declared with the **modifier public**, in which case that class is **visible to all classes everywhere**.

If a class has **no modifier** (the default, also known as *package-private*), it is **visible only within its own package** (packages are named groups of related classes)

Static binding (or early binding)

- Static binding: bind at compilation time
- Performed if the compiler can resolve the binding at compile time
- Applied for
 - Static methods
 - Private methods
 - Final methods
 - Fields

דריסה של שירותים

```
public class A {  
    public void print() {  
        System.out.println("A");  
    }  
}
```

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

הפלט:

B
B

```
public class C {  
    public static void main(String[]  
        args){  
        B b = new B();  
        A a = b;  
  
        b.print();  
        a.print();  
    }  
}
```

אין צורך ב-
casting

האם הקוד מתקמפל? אם לא, למה?
אם כן, האם יש שגיאת ריצה? אם יש, למה?
אחרת, מה הפלט?

דריסה של שירותים וניראות

```
public class A {  
    public void print() {  
        System.out.println("A");  
    }  
}
```

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

שגיאת קומפילציה:
"Cannot reduce the
visibility of the
inherited method from A"

```
public class C {  
    public static void main(String[]  
        args) {  
        B b = new B();  
        b.print();  
    }  
}
```

האם הקוד מתקמפל? אם לא, למה?
אם כן, האם יש שגיאת ריצה? אם יש, למה?
אחרת, מה הפלט?

דריסה של שירותים וניראות (2)

```
public class A {  
    protected void print() {  
        System.out.println("A");  
    }  
}
```

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

```
public class C {  
    public static void main(String[]  
        args) {  
        B b = new B();  
        b.print();  
    }  
}
```

הפלט:

B

האם הקוד מתקמפל? אם לא, למה?
אם כן, האם יש שגיאת ריצה? אם יש, למה?
אחרת, מה הפלט?

הורשה

```
public class A {  
    public void foo() {  
        System.out.println("A.foo()");  
    }  
  
    public void bar() {  
        System.out.println("A.bar()");  
        foo();  
    }  
}
```

```
public class B extends A {  
    public void foo() {  
        System.out.println("B.foo()");  
    }  
  
    public static void main(String[]  
        args) {  
        A a = new B();  
        a.bar();  
    }  
}
```

הפלט:

```
A.bar()  
B.foo()
```

האם הקוד מתקמפל? אם לא, למה?
אם כן, האם יש שגיאת ריצה? אם יש, למה?
אחרת, מה הפלט?

הורשה (2)

```
public class A {  
    private void foo() {  
        System.out.println("A.foo()");  
    }  
  
    public void bar() {  
        System.out.println("A.bar()");  
        foo();  
    }  
}
```

```
public class B extends A {  
    public void foo() {  
        System.out.println("B.foo()");  
    }  
  
    public static void main(String[]  
        args) {  
        A a = new B();  
        a.bar();  
    }  
}
```

הפלט:
A.bar()
A.foo()

האם הקוד מתקמפל? אם לא, למה?
אם כן, האם יש שגיאת ריצה? אם יש, למה?
אחרת, מה הפלט?

סדר הפעולות ביצירת אובייקט

- אתחול ערך דיפולטי לשדות מופע.
- קריאה לבנאי של מחלקת האב (שגורר אותו סדר פעולות רקורסיבית).
- אתחול שדות מופע לפי הערכים שהושמו להם בשורה שבה הם מוגדרים.
- ביצוע שאר הקוד של הבנאי.

סדר הפעולות ביצירת אובייקט

```
public class B extends A{  
    String bar = "B.bar";  
    B(){  
        bar = null;  
        super();  
        bar = "B.bar";  
        foo();  
    }  
}
```

סדר הפעולות ביצירת אובייקט

1. אתחול ערך דיפולטי לשדות מופע.
2. קריאה לבנאי של מחלקת האב (שגורר אותו סדר פעולות רקורסיבית).
3. אתחול שדות מופע לפי הערכים שהושמו להם בשורה שבה הם מוגדרים.
4. ביצוע שאר הקוד של הבנאי.

הורשה ובנאים

```
public class A {  
    String bar = "A.bar";  
  
    A() { foo(); }  
  
    public void foo() {  
        System.out.println("A.foo(): bar = " +  
            bar);  
    }  
}
```

```
public class B extends A {  
    String bar = "B.bar";  
  
    B() { foo(); }  
  
    public void foo() {  
        System.out.println("B.foo(): bar = " +  
            bar);  
    }  
}
```

```
public class C {  
    public static void main(String[]  
        args) {  
        A a = new B();  
        System.out.println("a.bar = "  
            + a.bar);  
        a.foo();  
    }  
}
```

מה פלט התוכנית?

```
B.foo(): bar = null  
B.foo(): bar = B.bar  
a.bar = A.bar  
B.foo(): bar = B.bar
```

הורשה ובנאים (2)

```
A {  
    public class A {  
        protected B b = new B();  
        public A() { System.out.println("in A: no args."); }  
        public A(String s) { System.out.println("in A: s = " + s); }  
    }  
}
```

```
B {  
    public class B {  
        public B() { System.out.println("in B: no args."); }  
    }  
}
```

```
C {  
    public class C extends A {  
        protected B b;  
        public C() { System.out.println("in C: no args."); }  
        public C(String s) { System.out.println("in C: s = " + s); }  
    }  
}
```

```
public class D {  
    public static void main(String args[]) {  
        C c = new C();  
        A a = new C();  
    }  
}
```

הפלט:

```
in B: no args.  
in A: no args.  
in C: no args.  
in B: no args.  
in A: no args.  
in C: no args.
```

הורשה ובנאים (3)

```
public class A {
    protected B b = new B();
    public A() { System.out.println("in A: no args."); }
    public A(String s) { System.out.println("in A: s = " + s); }
}

public class B {
    public B() { System.out.println("in B: no args."); }
}

public class C extends A {
    protected B b;
    public C() { System.out.println("in C: no args."); }
    public C(String s) { System.out.println("in C: s = " + s); }
}

public class D {
    public static void main(String args[]) {
        C c = new C("c");
        A a = new C("a");
    }
}
```

הפלט:

```
in B: no args.
in A: no args.
in C: s = c
in B: no args.
in A: no args.
in C: s = a
```

דריסה והעמסה של שירותים

```
public class A {  
    public float foo(float a, float b) throws IOException {  
    }  
}
```

```
public class B extends A {  
    ...  
}
```

אילו מהשירותים הבאים ניתן להגדיר ב-B?

- 1. `float foo(float a, float b){...}`
- 2. `public int foo(int a, int b) throws Exception{...}`
- 3. `public float foo(float a, float b) throws Exception{...}`
- 4. `public float foo(float p, float q) {...}`

הורשה ודריסת שירותים

```
public class A {  
    public void foo() {...}  
}
```

האם אפשר לקרוא ל-foo של
A מתוך B?

```
public class B extends A {  
    public void foo() {...}  
}
```

תשובה:
דרך `super.foo()`

הורשה ודריסת שירותים (2)

```
public class A {  
    public void foo() {...}  
}
```

האם אפשר לקרוא ל-foo של
A מתוך C?

```
public class B extends A {  
    public void foo() {...}  
}
```

תשובה:
אי אפשר,
`super.super.foo()`
- לא חוקי

```
public class C extends B {  
    public void foo() {...}  
}
```

מחלקות פנימיות

```
public class Test {  
    public int a = 0;  
    private int b = 1;
```

אילו משתנים מ-a-e נגישים מהשורה המסומנת?

```
    public void foo(final int c) {  
        int d = 2;
```

תשובה: כולם חוץ מ-d

```
        class InnerTest {  
            private void bar(int e) {
```

```
            }
```

```
        }
```

```
        d = 3;
```

```
        a = 3;
```

```
    }
```

```
}
```


מחלקות פנימיות - סיכום

Type	Scope	Inner	Interface	Fields access
Static nested	member	no	yes	Only static
Inner non-static	member	yes	no	Static and non-static
local	Local scope	yes	no	Effectively final local variables or parameters that are accessible in the scope of the block
anonymous	Only the point where it is defined	yes	no	Effectively final local variables or parameters that are accessible in the scope of the block