

תוכנה 1

תרגול 14 – סיכום

קצת על ממשקים

■ ממשק יכול להרחיב יותר מממשק אחד

■ שירותים בממשק הם תמיד מופשטים וציבוריים

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

The “type” of foo1 and foo2 is 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?

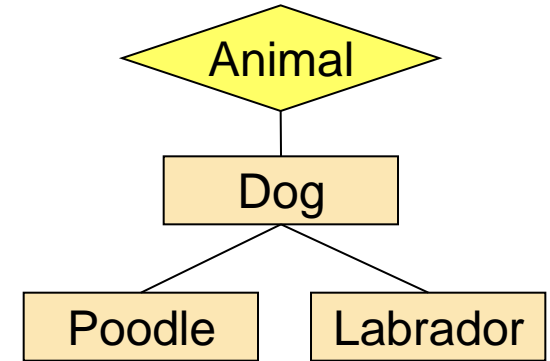
פלט:
"No exception is thrown"



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;
```

poodle = (Poodle) dog;
-No compilation error
-Runtime Exception

- Compilation Error
Type mismatch: cannot convert

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

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) {...}`

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?

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() {...}  
}
```

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();
    }
}
```

הפלט:

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


הורשה ובנאים (2)

```
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();
        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
```

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) {  
                  
            }  
        }  
    }  
}
```

בחינה באופק!

- הבחינה תכלול את כל הנושאים שסינינו במהלך הסמסטר (שיעורים, תרגולים ועבודות בית)
- Java, DBC, ירושה ופולימורפיזם, IO, iterator, ... , Generics, מחלקות פנימיות, Collection Framework
- לפתור כמה שיותר מבחנים משנים שעברו
- לא כל הסמסטרים זהים מבחינת חומר
- לתרגל כתיבת קוד על נייר

בהצלחה!