Typer

Joachim von Hacht

Vad kan hända med Typer?

Kompileringsfel (typfel):

- Inte super/sub vid tilldelning
- Felaktig operation t.ex. [] på icke-array eller ++ på icke numeriskt typ.
- Instansiera gränssnitt eller abstrakt klass (new).

Körningsfel:

- Explicit typomvandling, objektet kan inte det (saknar metoder) som typen för variabeln kräver (ClassCastException)
- Stoppa in i referens-array (hål i typsystemet, ArrayStoreException)

Vilken metod körs?

- Override kräver att arv är iblandat. Flera klasser i arvshierakin har exakt samma metod. Objektets typ under körning avgör vilken metod som körs.
- Overload, flera metoder med samma namn men olika parametrar. Bestäms (fixeras) vid kompilering utifrån variabelns typ (implicit typomvandling av argument kan förekomma, metoden kan vara ärvd)

Typer (1)

Vilka rader ger kompileringsfel (char <: int <: double)?

```
int i = 1;
char ch = '1';
double d = 1.0;

i = d;                  // 1
i = ch;                 // 2
ch = i;                 // 3
ch = d;                 // 4
d = i;                  // 5
d = ch;                 // 6
i = (int) d + 0.1;      // 7
(double) i = d;         // 8
```

Typer (2)

Vilka rader ger kompileringsfel? Vilka ger körningsfel?
(Integer <: Object och Double <: Object)

```
Integer[] is = {1, 2, 3};
Double[] ds = {1.0, 2.0, 3.0};
Object o = is;                          // 1
o[0]++;                                 // 2

Object[] os1 = is;                      // 3
os1[0]++;                               // 4

Object[] os2 = (Object[]) is;           // 5
Double[] ds1 = (Double[]) os1;          // 6

os2 = ds;                               // 7
Double[] ds2 = (Double[]) os2;          // 8

Integer i1 = os1[0];                    // 9
Integer i2 = (Integer) os1[0];          // 10
```

Typer (3)

Vilka rader ger kompileringsfel? Vilka ger körningsfel?

```
A a = new A();      // 1
B b = new B();      // 2
IX ix = new IX();   // 3
IY iy = null;       // 4

a = b;              // 5
b = (A) a;          // 6

ix = iy;            // 7
iy = ix;            // 8

ix = a;             // 9
a = (A) ix;         // 10
ix = b;             // 11
iy = b;             // 12

ix = (IX) iy;       // 13
```

```
interface IX {
  ...
}

interface IY {
   ...
}

class A implements IX {
   ...
}

class B implements IY {
   ...
}
```

Typer (4)

Vilka rader ger kompilerings resp. körningsfel. Varför? Om fel åtgärdas vad skrivs ut?

```
A a = new A();
B b = new B();

IX ix = a;     // 1
IY iy = a;     // 2
a.doIt();      // 3
ix.doIt();     // 4
iy.doIt();     // 5
iy.doOther();  // 6


iy = b;          // 7
iy = (IY) b;     // 8
iy.doOther();    // 9

iy = new C();    // 10
iy.doOther();    // 11
```

```
interface IX { void doIt();}
interface IY { void doOther();}

class A implements IX, IY {
   public void doIt() {
       out.println("A doIt()");
   }
   public void doOther() {
       out.println("A doOther()");
   }
}
class B {
   public void doOther() {
        out.println("B doOther()");
   }
}
class C implements IY {
   public void doOther() {
      out.println("C doOther()"); }
}
```

Typer (5)

Vilka exempel ger kompilerings resp. körningsfel. Varför? Om fel åtgärdas vad skrivs ut?

```
//a
A a = new B();
a.doA();


//b
IA a = new X();
a.doA();

//c
C c = new B();
c.doC();

//d
B b1 = new C();
b1.doX();

//e
IX x = new C();
X x1 = (X) x;
x1.doA();

//f
IX x2 = new C();
B b2 = (B) x2;
b2.doC();
```

```
public interface IA {void doA();}
public interface IX {void doX();}

public abstract class A implements IA {
   public void doA() {out.println("A.doA()");}
   public abstract void doC();
}

public class B extends A {
   public void doC() {out.println("B.doC()");}
}

public class C extends B implements IX {
   public void doA() {out.println("C.doA()");}
   public void doX() {out.println("C.doX()");}
   public void doC() {out.println("C.doC()");}
}

public class X implements IX {
   public void doX() {out.println("X.doX()");}
   public void doA() {out.println("X.doA()");}
}
```

Vad skrivs ut?

Kompilerar detta? Fungerar det att köra?
Alla typer <: Object

```
Object[] o = { 0, "abc", 1.23 };
for (int i = 0; i < o.length; i++) {
    classify(o[i]);
}

void classify(Integer i) {
   out.println("It's an integer");
}

void classify(String s) {
   out.println("It's a String");
}

void classify(Object o) {
   out.println("Don't know, it's anything...");
}
```

Vad händer?

Kompilerar detta? Får vi något körningsfel?

```
void program() {
   Integer[] arr = {1, 2, 3};
   upDate(arr);
}

void upDate(Object[] arr) {
   a[0] = null;
   a[1] = 1;
   a[2] = 1.0;
}
```