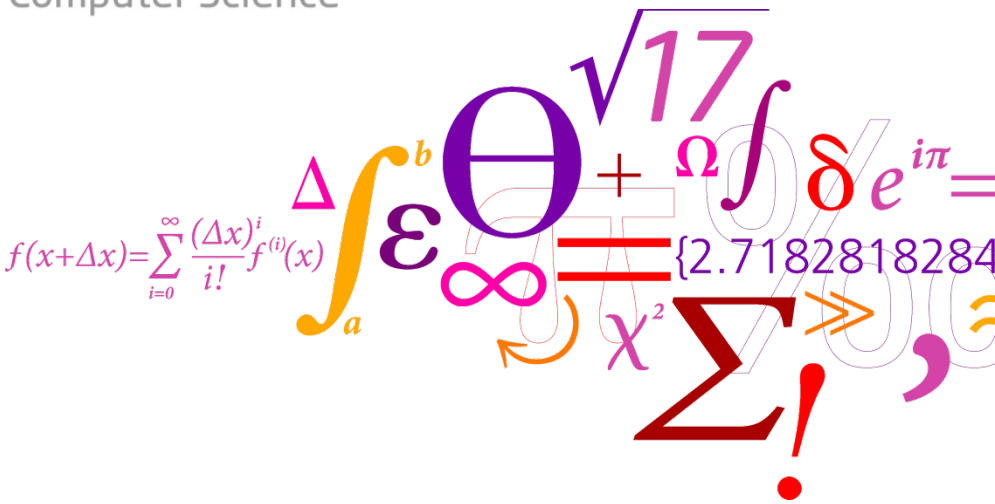
Slide PiSU-L07.1

Projekt i software-udvikling (02362)
Forår 2020

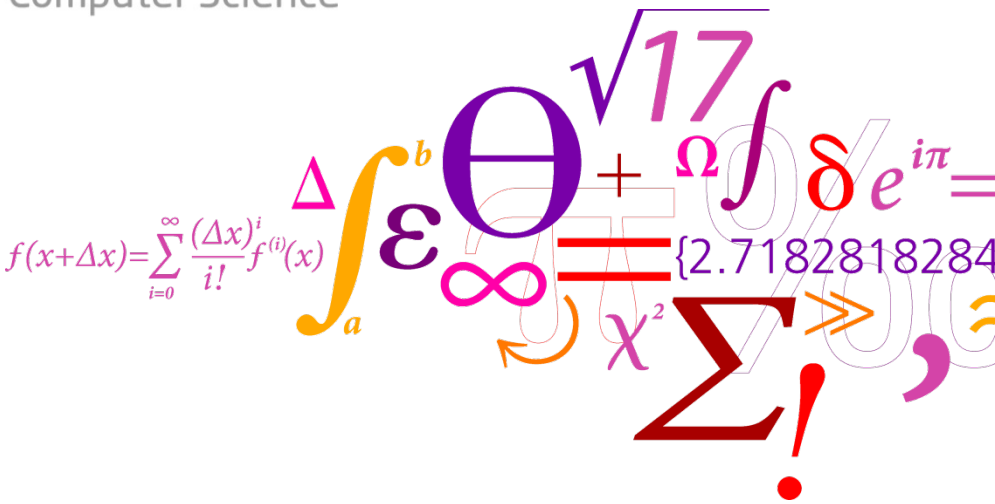
Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



IV. Java Praksis

DTU Compute
Department of Applied Mathematics and Computer Science



4. Undtagelser (Exceptions)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Nogle gange kommer det frem midt i en metode, at operationen ikke kan gennemføres. Så er man nødt til at reagere og også den som har kaldt metoden er måske nødt til at reagere på det.

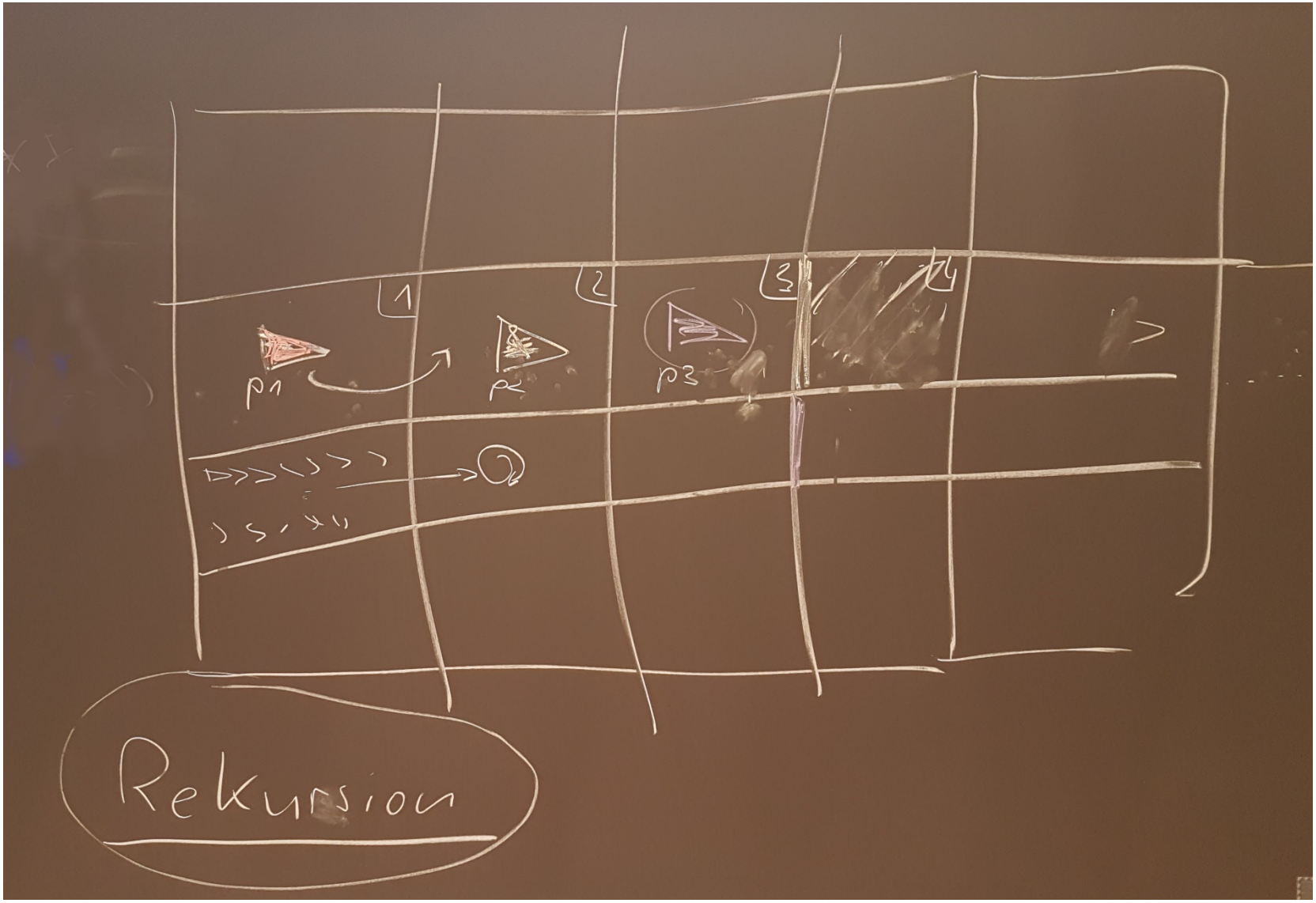
Såkaldte undtagelser (exceptions) er en mulighed at programmere sådan noget.

Eksempel: Flyt en robot

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Hvis en robot skal flyttes, skubber den en robot som star foran den også I same retning; og hvis der star en foran den igen, skal den også flyttes, osv.
- Det giver en kaskade af flytningerne
- Hvis så den sidste i kaskaden ikke kan flyttes (fx da den rammer en væg), så ryger hele kaskaden, og den oprindelige træk er ikke muligt

→ Se diskussion på tavlen!


1
2
3
4
p1
p2
p3
Rekursion

Deklaration af Exception

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class ImpossibleMoveException extends Exception {

    private Player player;
    private Space space;
    private Heading heading;

    public ImpossibleMoveException(Player player,
            Space space,
            Heading heading) {
        super("Move impossible");
        this.player = player;
        this.space = space;
        this.heading = heading;
    }
}
```

Bliver ikke brugt endnu!

MoveForward()

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public void moveForward(@NotNull Player player) {
    if (player.board == board) {
        Space space = player.getSpace();
        Heading heading = player.getHeading();

        Space target = board.getNeighbour(space, heading);
        if (target != null) {
            try {
                moveToSpace(player, target, heading);
            } catch (ImpossibleMoveException e) {
                // we don't do anything here  for now;
                // we just catch the exception so that
                // we do no pass it on to the caller
                // (which would be very bad style).
}   }   }   }
```

Skubbe-metoden

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
private void moveToSpace(
        @NotNull Player player,
        @NotNull Space space,
        @NotNull Heading heading) throws ImpossibleMoveException {

    Player other = space.getPlayer();
    if (other != null){
        Space target = board.getNeighbour(space, heading);
        if (target != null) {
            // XXX Note that there might be additional problems
            //     with infinite recursion here!
            moveToSpace(other, target, heading);
        } else {
            throw new ImpossibleMoveException(player, space, heading);
        }
    }
    player.setSpace(space);
}
```

Kontrolflow. Exceptions


```
try {
   ...
   do something;
   callSomeone();
   ...
} catch (Expeption1 e) {
   ...
   do something
} catch (Expeption2 e) {
   ...
   do something
} finally {
   ...
   do something
}
```

Hvis der ikke sker en exception i try-blokken,

Så bliver catch-blokkene sprunget over!

Men finally-bokken (hvis den er der) bliver ekseveret

Kontrolflow:Exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
try {
   ...
   do something;
   callSomeone();
   ...
} catch (Exeption1 e) {
   ...
   do something
} catch (Exeption2 e) {
   ...
   do something
} finally {
   ...
   do something
}
```

Når der sker en exception i try-blokken (som ikke bliver fanget indeni),

bliver resten af koden i try-blokken sprunget over

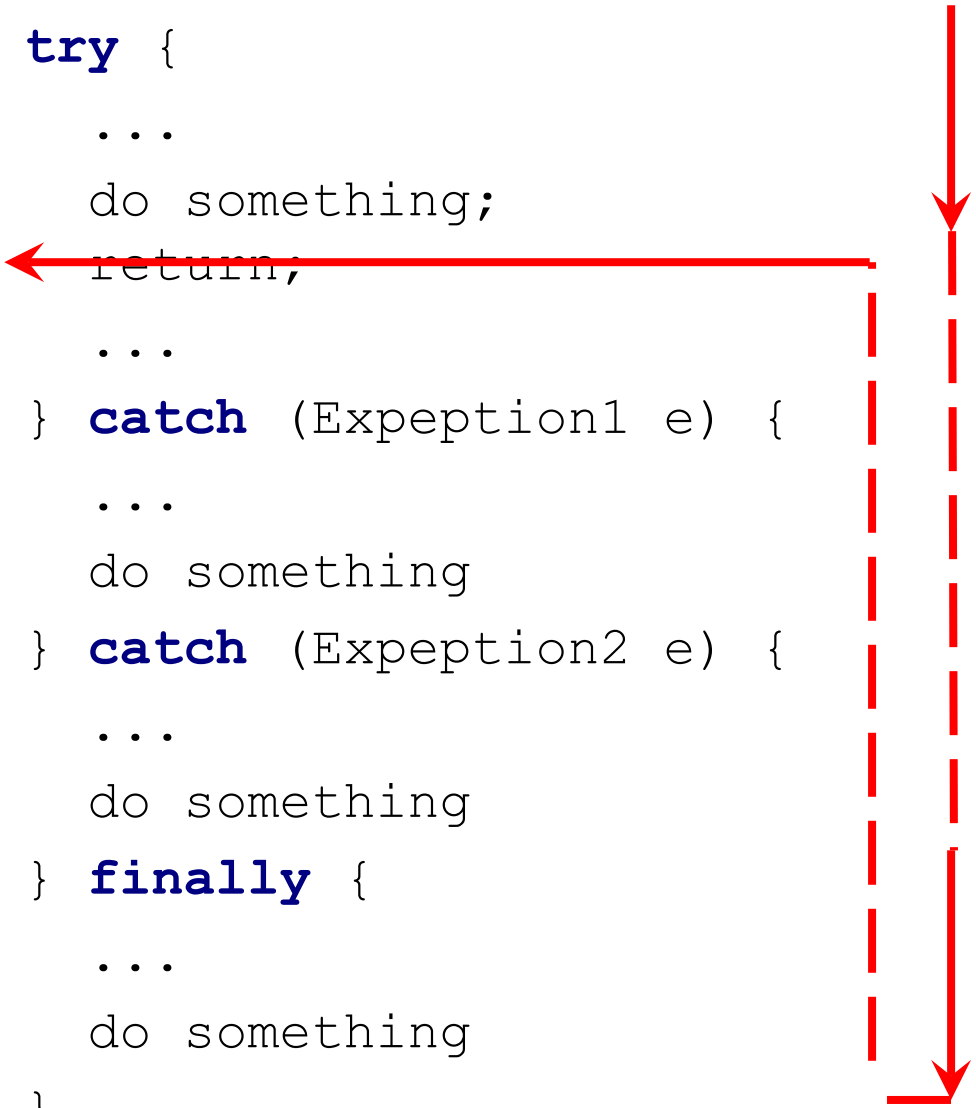
og den første catch-blok som matcher (er super klasse af) den udløste exception eksekveret (men kun en!)

og finally-bokken (hvis den er der) bliver også ekseveret!

En krølle: finally/return

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
try {
   ...
   do something;
   return;
   ...
} catch (Exeption1 e) {
   ...
   do something
} catch (Exeption2 e) {
   ...
   do something
} finally {
   ...
   do something
}
```



Finally-blokken bliver altid eksekveret, hvis kontrollen er i en try-blok eksekveret!