Activity Diagrams

and their analogies in code (TypeScript)

Prof. Dipl.-Ing. Jirka R. Dell'Oro-Friedl

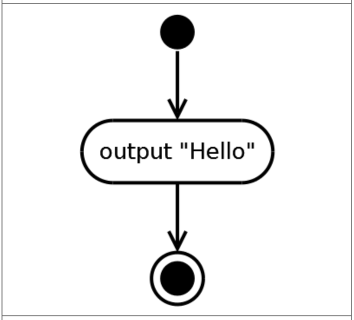
V1.0 ©HFU2018

1. Elements

●	Startknoten (Initial Node)
◉	Endknoten (ActivityFinalNode)
⊗	Ablaufendknoten (FlowFinalNode)
Ⓐ	Konnektor (Connector)
◇	Entscheidung und Zusammenführung (DecisionNode / MergeNode)
▬	Teilung / Synchronisation (ForkNode / JoinNode)
⋔	Aufruf
↗ [Bedingung]	Kontrollfluss / Objektfluss ActivityEdge (ControlFlow / ObjectFlow)
(Aktivität)	Aktionsknoten (ActionNode)
[Daten]	Objektknoten (ObjectNode)
Notiz	Notiz (Note)
⧗	Zeitsignal (AcceptTimeEventAction)
Event	Signalempfang (AcceptEventAction)
Event	Signalversand (SendSignalAction)

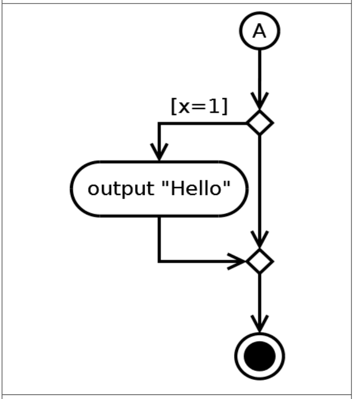
2. Basic flow structures

Linear



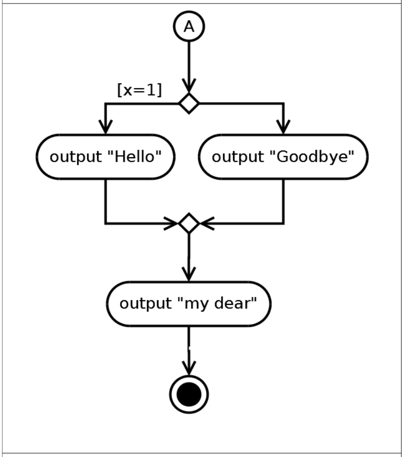
```
console.log("Hello");
```

Conditional



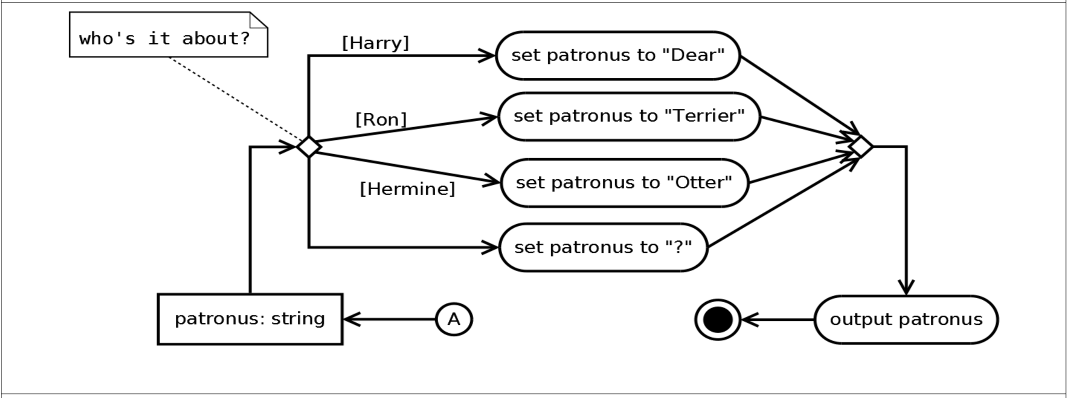
```
...
if (x == 1)
 console.log("Hello");
```

Exclusive Conditional



```
...
if (x == 1)
  console.log("Hello");
else
  console.log("Goodbye");
console.log(" my dear");
```

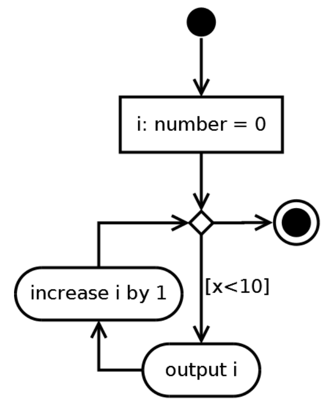
Multiple Conditions



```
...
let patronus: string;
switch (person) {
    case "Harry":
        patronus = "Dear";
        break;
    case "Hermine":
        patronus = "Otter";
        break;
    case "Ron":
        patronus = "Terrier";
        break;
    default:
        patronus = "?";
        break;
}
console.log(patronus);
```

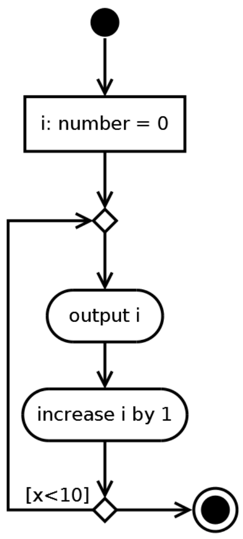
3. Loops

Loop (Pre-Test)



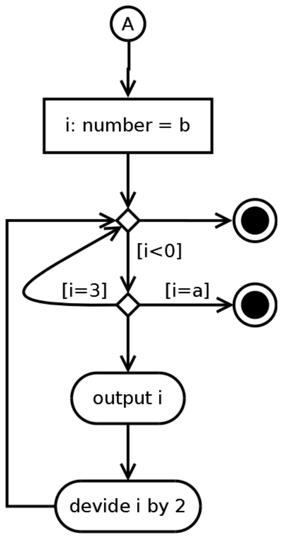
```
let i: number = 0;
while (i < 10) {
    console.log(i);
    i++;
}
                oder

for (let i: number = 0; i < 10; i++)
    console.log(i);
```

Loop (Post-Test)



```
let i: number = 0;
do {
    console.log(i);
    i++;
} while (i < 10);
```

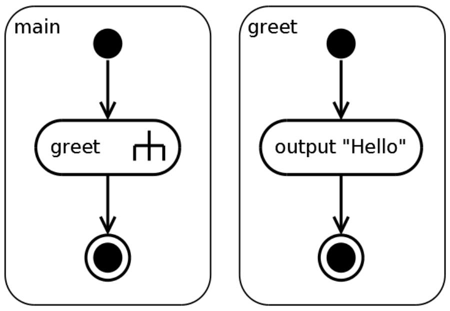
Loop with additional control conditions



```
for (let i: number = b; i > 1; i/=2)
{
    if (i == 3)
        continue;
    if (i == a)
        break;
    console.log(i);
}
```

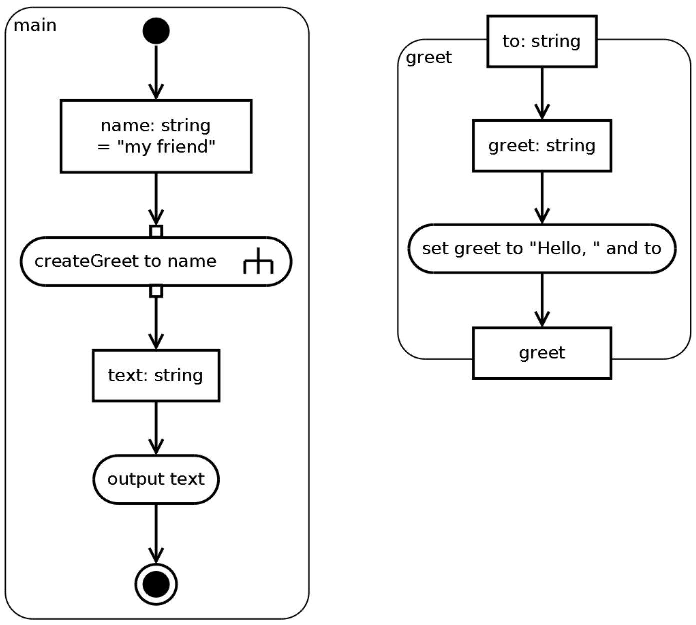
4. Subactivities

Subactivity (Function/Method)



```
greet();
function greet(): void {
    console.log("Hello");
}
```

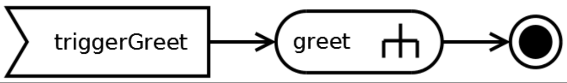
Subactivity with Input/Output



```
let name: string = "my friend";
let text: string = createGreet(name);
console.log(text);

function createGreet(_to: string): string {
    let greet: string;
    greet = "Hello, " + _to;
    return greet;
}
```

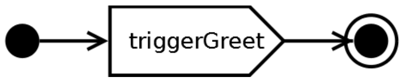
5. Signals

Accept Event



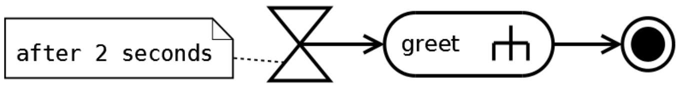
```
window.addEventListener("triggerGreet", greet);
```

Send Signal



```
let event: Event = new Event("triggerGreet");
window.dispatchEvent(event);
```

Accept Time Event



```
window.setTimeout(greet, 2000);
```