TDP002 - Imperativ Programmering

Abstrakta Datatyper

Pontus Haglund

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

1. Mål med föreläsning(arna)
2. Grundläggande ADT
3. Bygga ut en ADT
4. Respektera Gränssnittet
5. Design

1 Mål med föreläsning(arna)
2 Grundläggande ADT
3 Bygga ut en ADT
4 Respektera Gränssnittet
5 Design

Mål med föreläsning(arna)

Efter föreläsningen skall studenten:

- Förstå grunderna för en ADT
- Förstå skillnaden mellan programmeraren och användaren av en ADT
- Ha en inledande förståelse för inkapsling
- Kunna designa och implementera enkla ADT

Tips: Skapa en Huvudprogramsfunktion

```
def factorial():
    total = 1
    for i in range(1, n+1):
        total *= i
    return total

n = 5
result = factorial()
print(result)
```

Tips: Skapa en Huvudprogramsfunktion

```
def factorial():
    total = 1
    for i in range(1, n+1):
        total *= i
    return total

def main():
    n = 5
    result = factorial()
    print(result)

if __name__ == "__main__":
    main()
```

Tips: Skapa en Huvudprogramsfunktion

```
def factorial(n):
    total = 1
    for i in range(1, n+1):
        total *= i
    return total

def main():
    n = 5
    result = factorial(n)
    print(result)

if __name__ == "__main__":
    main()
```

1 Mål med föreläsning(arna)
2 Grundläggande ADT
3 Bygga ut en ADT
4 Respektera Gränssnittet
5 Design

Grundläggande ADT

Varför ADT?

- Finns många saker vi vill representera i ett program
- Finns ett begränsat antal datatyper
- Vi kan (utan särskilt stöd i språket) skapa egna ADT
- Skapa ett lager av abstraktion för att göra det lättare

Grundläggande ADT

Hur ADT?

- En samling av funktioner (agerar gränssnitt)
- En underliggande (okänd) behållare
- Dessa utgör tillsammans en abstraktion

Grundläggande ADT

Användare och Programmerare

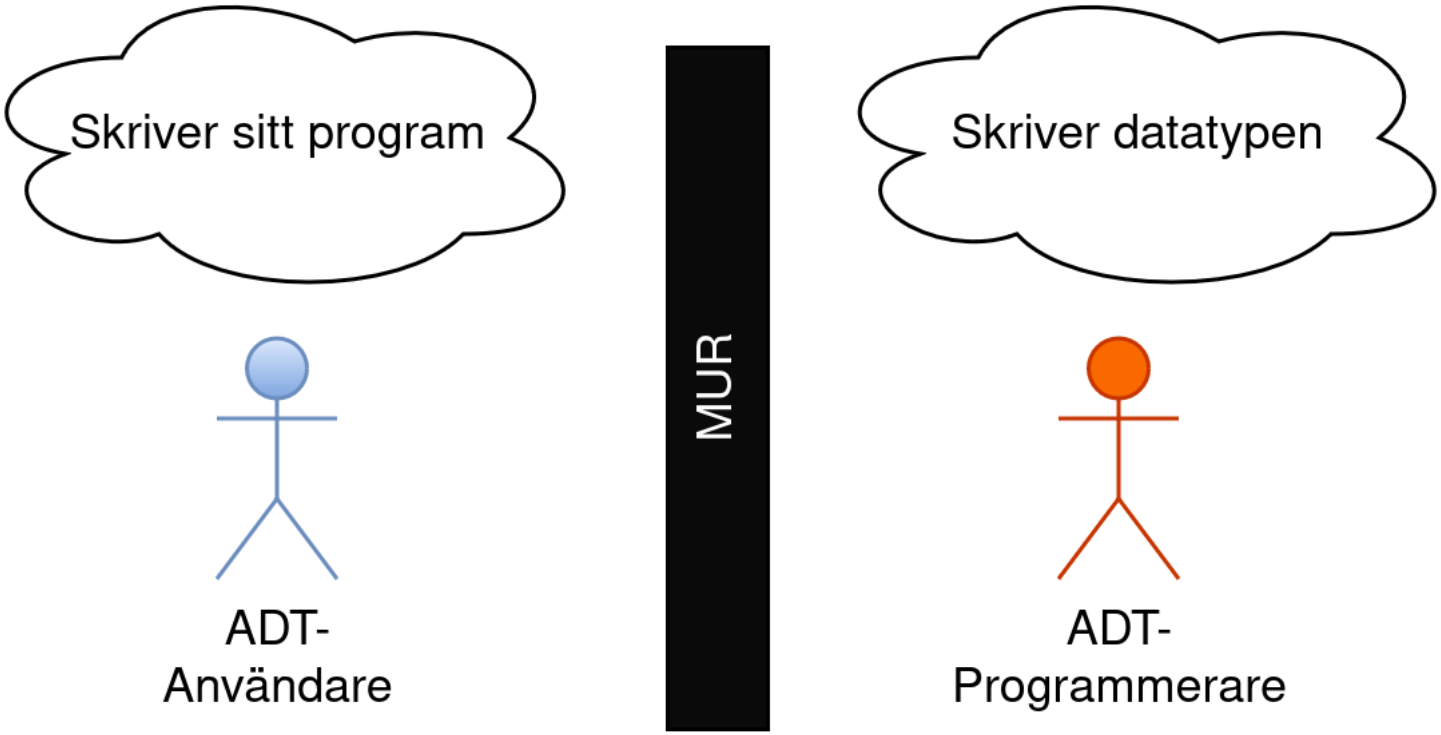
In computer science, an abstract data type (ADT) is a mathematical model for data types, defined by its behavior (semantics) from the point of view of a user of the data, specifically in terms of possible values, possible operations on data of this type, and the behavior of these operations.
- Wikipedia Abstract data type

ADT-
Användare

ADT-
Programmerare

Grundläggande ADT

Användare och Programmerare



Grundläggande ADT

```
# Student ADT
def create(liuid):
    return {"liuid": liuid}

def display(student):
    print("liu-id:", student["liuid"])

# Main Program
student = create("ponha45")
display(student)
```

```
$ python3 student.py
> liu-id: ponha45
```

Hur Fungerar Inbyggda Typer?

```
results = [6, 2, 3, 1, 5]
results.append(0)
results.sort()
results.pop()
```

ADT-
Användare

Hur Fungerar Inbyggda Typer?

Svaret är att vi vet hur vi kan använda dem, men inte hur de fungerar bakom kulisserna. För att ta reda hur vi kan använda `List` tittar vi i dokumentationen.

Det är samma princip med en ADT, ADT-programmeraren vet hur det fungerar bakom kulisserna, men det gör **inte** ADT-användaren!

Notera att `List` är en datastruktur inte en ADT.

1 Mål med föreläsning(arna)
2 Grundläggande ADT
3 Bygga ut en ADT
4 Respektera Gränssnittet
5 Design

LINKÖPING UNIVERSITY

Bygga ut en ADT

Hur Användbar är denna ADT?

```
# Student ADT
def create(liuid):
    return {"liuid": liuid}

def display(student):
    print("liu-id:", student["liuid"])

# Main Program
student = create("ponha45")
display(student)
```

Bygga ut en ADT

Features

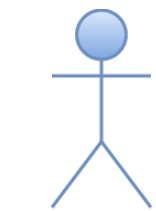
- Skapa en ny student
- Skriva ut en student
- Lägga till kurser
- Sätta betyg i kurser

```
student = create("ponha45")
display(student)
```

Bygga ut en ADT

Features

- Skapa en ny student
- Skriva ut en student
- Lägga till kurser
- Sätta betyg i kurser



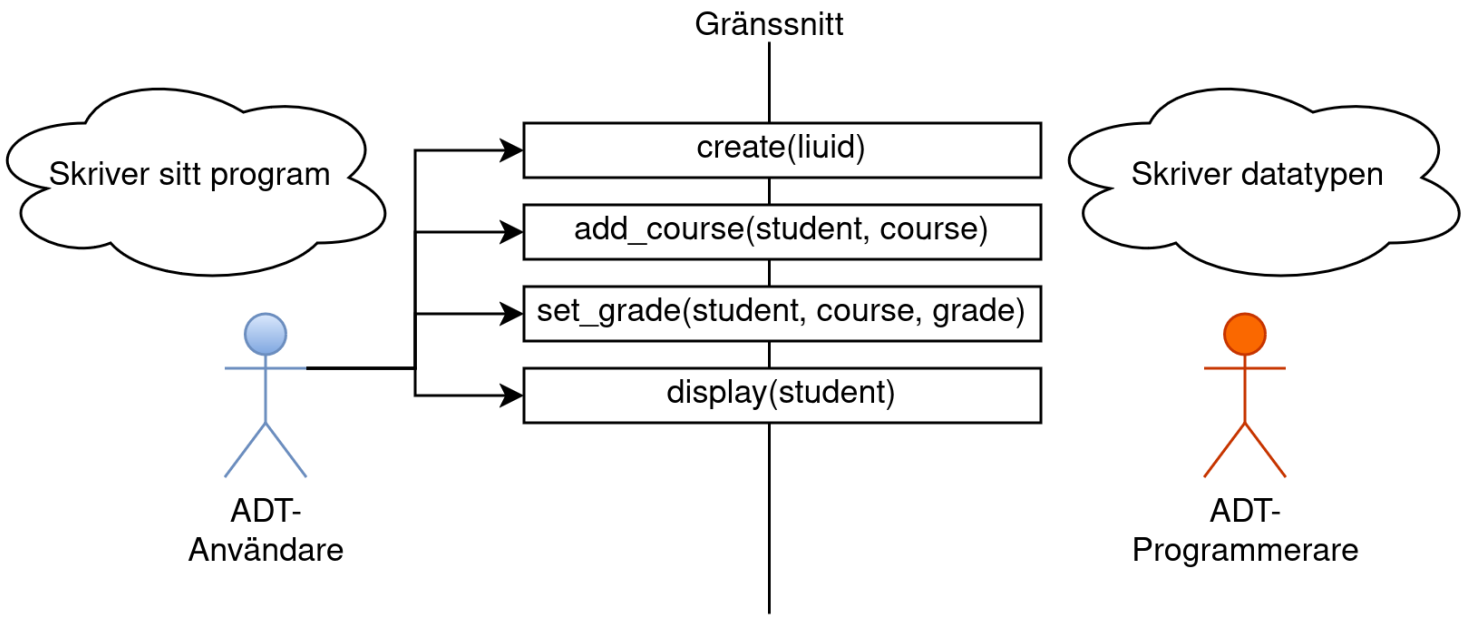
ADT-
Användare

```
student = create("ponha45")
add_course(student, "TDP001")
add_course(student, "TDP002")
add_course(student, "TDP003")
set_grade(student, "TDP001", 3)
display(student)
```

```
liu-id: ponha45
courses
TDP001: 3
TDP002: -
TDP003: -
```

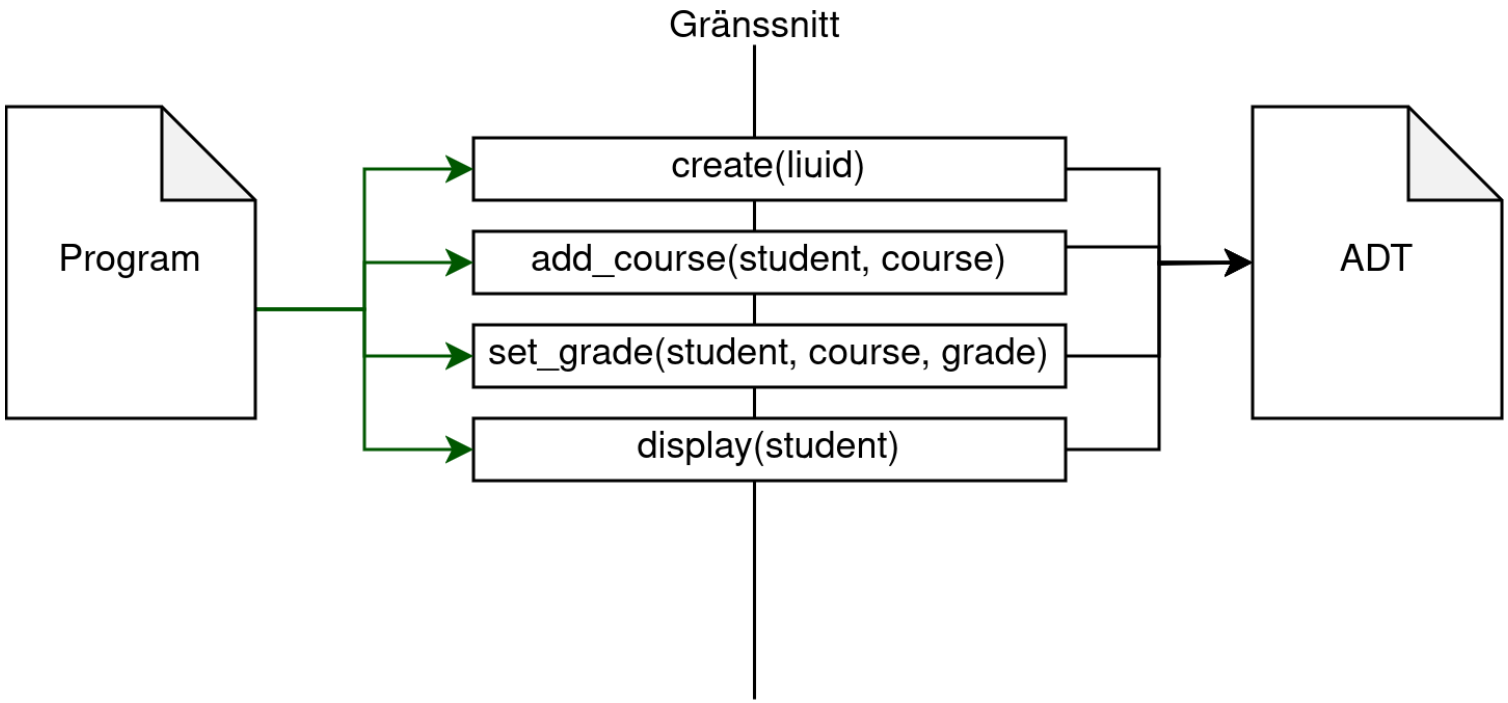
Bygga ut en ADT

Gränssnittet



Bygga ut en ADT

Gränssnittet cont.



Bygga ut en ADT

```
def create(liuid):
    return {"liuid": liuid, "courses": {}}

def add_course(student, course, grade="-"):
    student["courses"][course] = grade

def set_grade(student, course, grade):
    student["courses"][course] = grade

def display(student):
    print("liu-id:", student["liuid"])
    print("courses")
    for k, v in student["courses"].items():
        print(k, ": ", v, sep="")
```

1 Mål med föreläsning(arna)
2 Grundläggande ADT
3 Bygga ut en ADT
4 Respektera Gränssnittet
5 Design

LINKÖPING UNIVERSITY

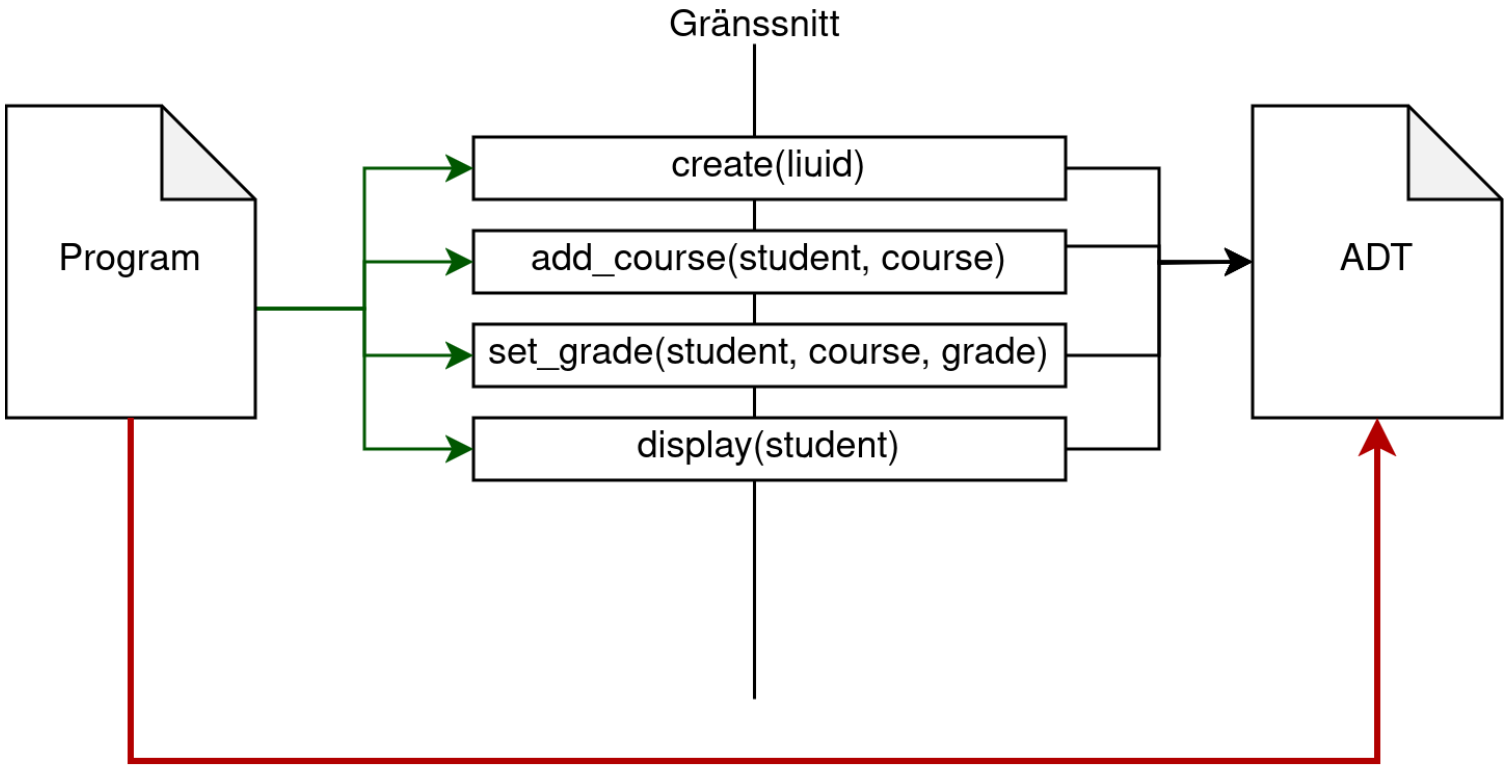
Respektera Gränssnittet

Bryta mot gränssnittet

```
student = create("ponha45")
add_course(student, "TDP001")
add_course(student, "TDP002")
add_course(student, "TDP003")
set_grade(student, "TDP001", 3)
student["liuid"] = "lovar08"
display(student)
```

Respektera Gränssnittet

Bryta mot gränssnittet cont.



Respektera Gränssnittet

Varför spelar detta någon roll?

Kan tyckas smidigt att ibland jobba med den interna representationen...

```
student = create("ponha45")
add_course(student, "TDP001")
add_course(student, "TDP002")
add_course(student, "TDP003")
set_grade(student, "TDP001", 3)
print(student)
```

```
{
  'liuid': 'ponha45',
  'courses': {
    'TDP001': 3,
    'TDP002': '-',
    'TDP003': '-'
  }
}
```

Respektera Gränssnittet

Varför spelar detta någon roll?

Det är viktigt att tänka på att den interna representationen inte är garanterad för användaren, bara gränssnittet.

```
student = create("ponha45")
add_course(student, "TDP001")
add_course(student, "TDP002")
add_course(student, "TDP003")
set_grade(student, "TDP001", 3)
print(student)
```

```
[
  'ponha45',
  {
    'TDP001': 3,
    'TDP002': '-',
    'TDP003': '-'
  }
]
```

Respektera Gränssnittet

```
def create(liuid): # Now with Lists
    return [liuid, {}]

def add_course(student, course, grade="-"):
    student[1][course] = grade

def set_grade(student, course, grade):
    student[1][course] = grade

def display(student):
    print("liu-id:", student[0])
    print("courses")
    for k, v in student[1].items():
        print(k, ": ", v, sep="")
```

Respektera Gränssnittet

Användarens program fungerar här

```
student = create("ponha45")
add_course(student, "TDP001")
add_course(student, "TDP002")
add_course(student, "TDP003")
set_grade(student, "TDP001", 3)
display(student)
```

```
liu-id: ponha45
courses
TDP001: 3
TDP002: -
TDP003: -
```

Respektera Gränssnittet

Användarens program fungerar inte här

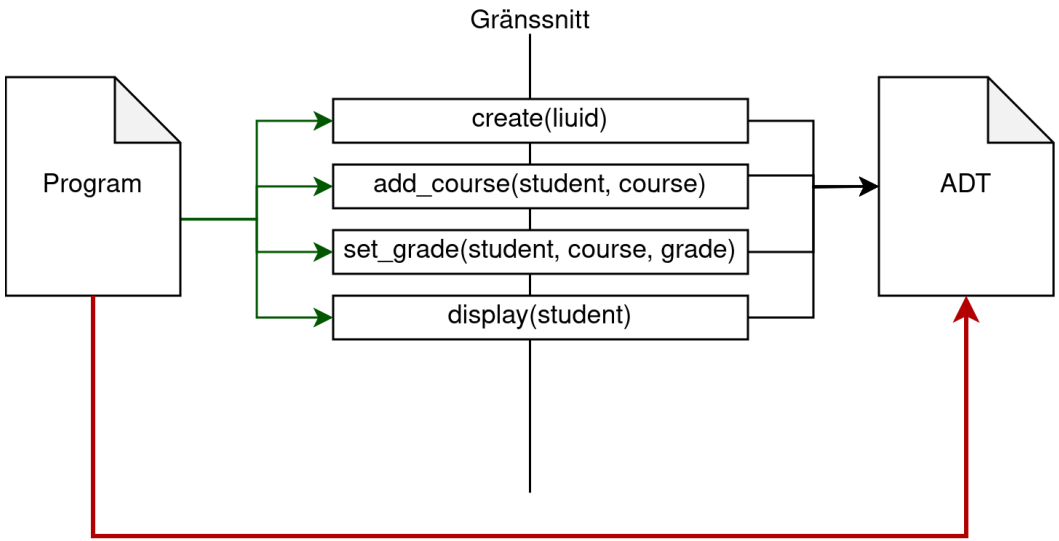
```
student = create("ponha45")
add_course(student, "TDP001")
add_course(student, "TDP002")
add_course(student, "TDP003")
set_grade(student, "TDP001", 3)
student["liuid"] = "lovar08"
display(student)
```

```
student["liuid"] = "lovar08"
TypeError: list indices must
be integers or slices, not str
```

Respektera Gränssnittet

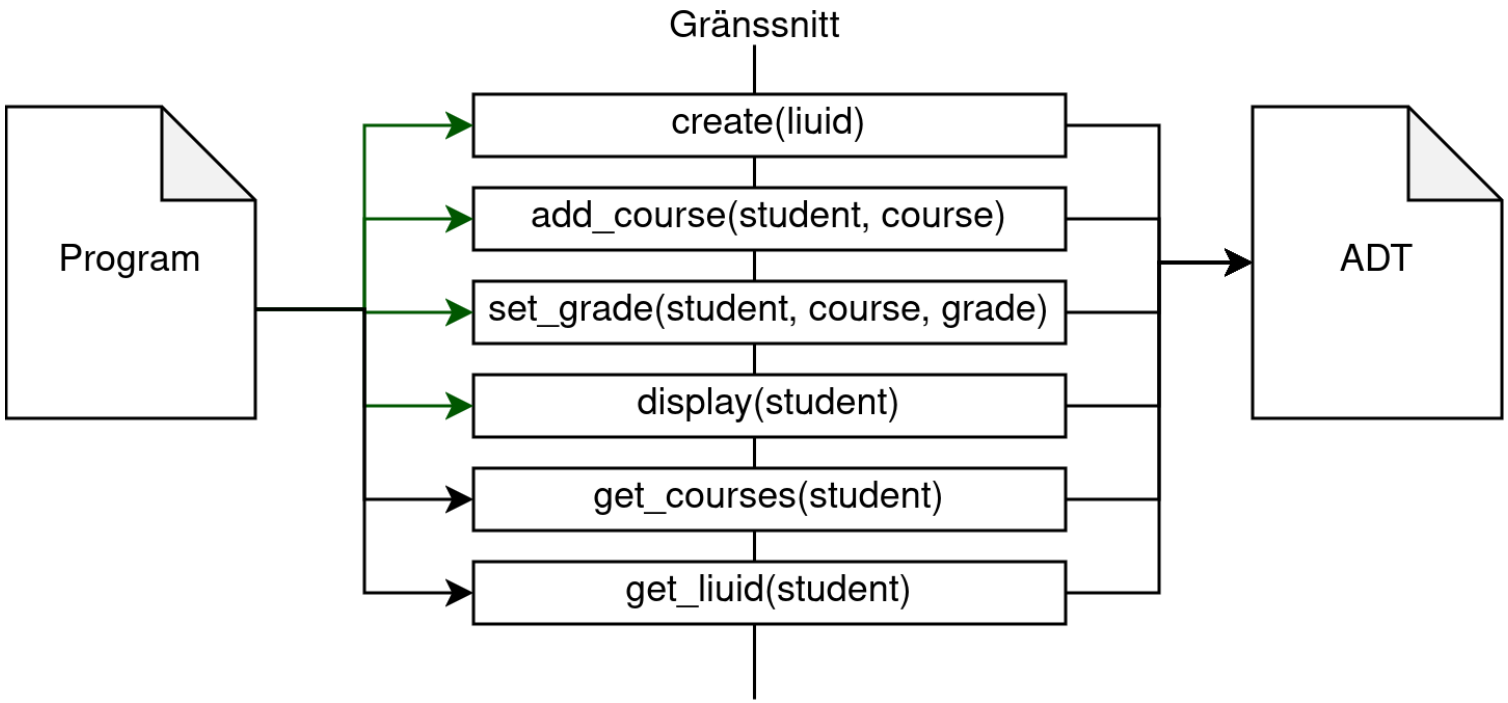
Men jag behöver kunna göra...

Jag kanske vill kunna göra något som saknas...



Respektera Gränssnittet

Utöka gränssnittet



Respektera Gränssnittet

Hur vet jag om jag bryter mot gränssnittet?

Om något svar på dessa frågor är sant så byter du förmodligen mot gränssnittet. Frågorna bör ställas till någon som (låtsas) inte har sett implementationen av ADT.

- Vet du vad det finns för underliggande behållare (`List`/`Dictionary`) i ADT?
- Används någon funktion för `List` eller `Dictionary` i programmet (e.g. `append`, `pop` eller `[]`)?

1 Mål med föreläsning(arna)
2 Grundläggande ADT
3 Bygga ut en ADT
4 Respektera Gränssnittet
5 Design

LINKÖPING UNIVERSITY

Design

Primitiver

- Konstruktorer: Skapar ett nytt objekt
- Selektorer: Hämtar ut data ur ett objekt
- Iteratorer: Går igenom flera delelement i ett objekt
- Modifikatorer: Förändrar data i ett objekt

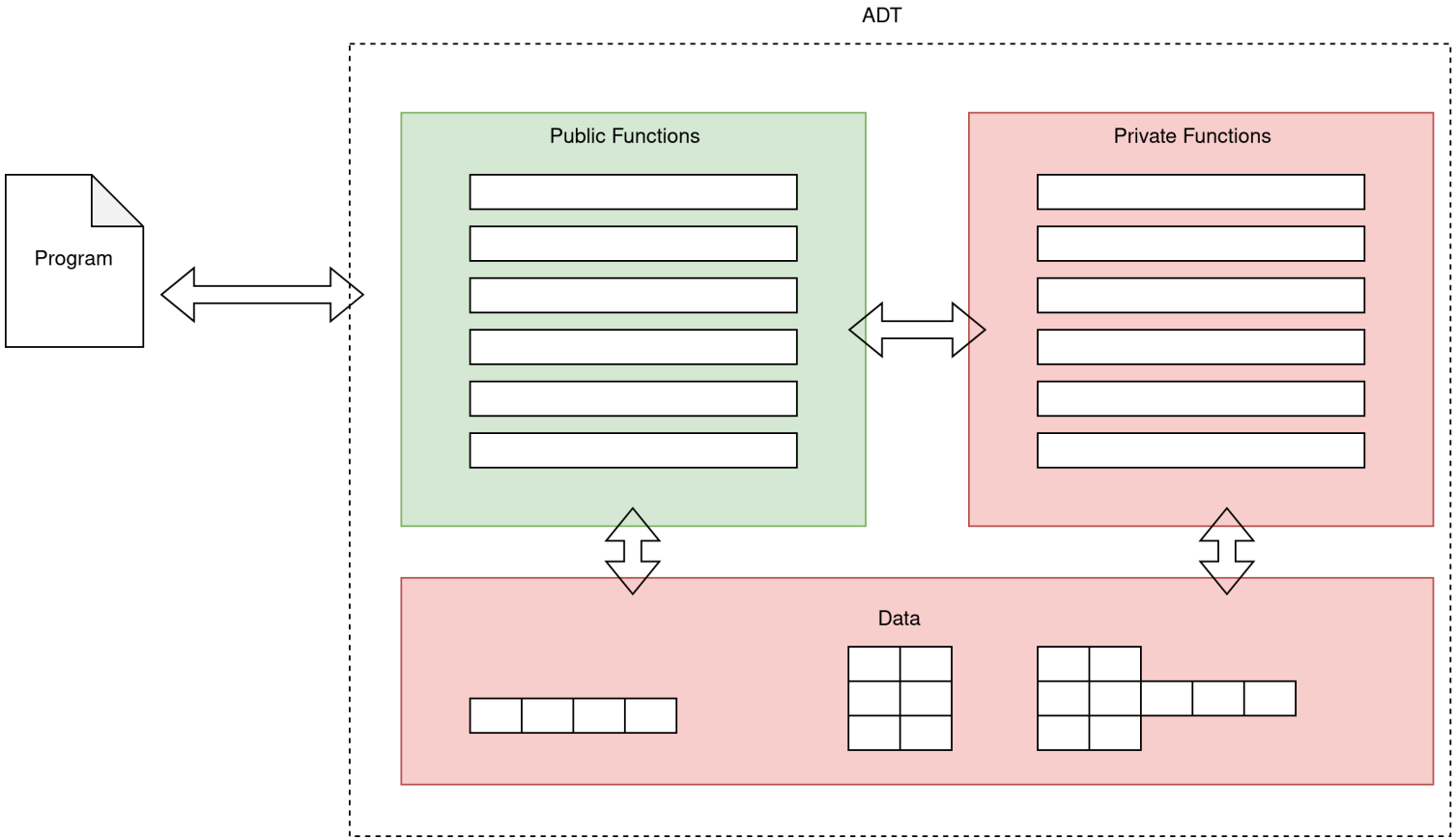
Design

Objekt och ADT

- En ADT är hela samlingen av funktioner m.m. (se slide i början med definitionen)
- Ett objekt är det som användaren av ett ADT får ut när konstruktorn körs
- I exemplet nedan är `student` objektet:

```
student = create("ponha45")
```

Design



www.liu.se

LINKÖPING UNIVERSITY