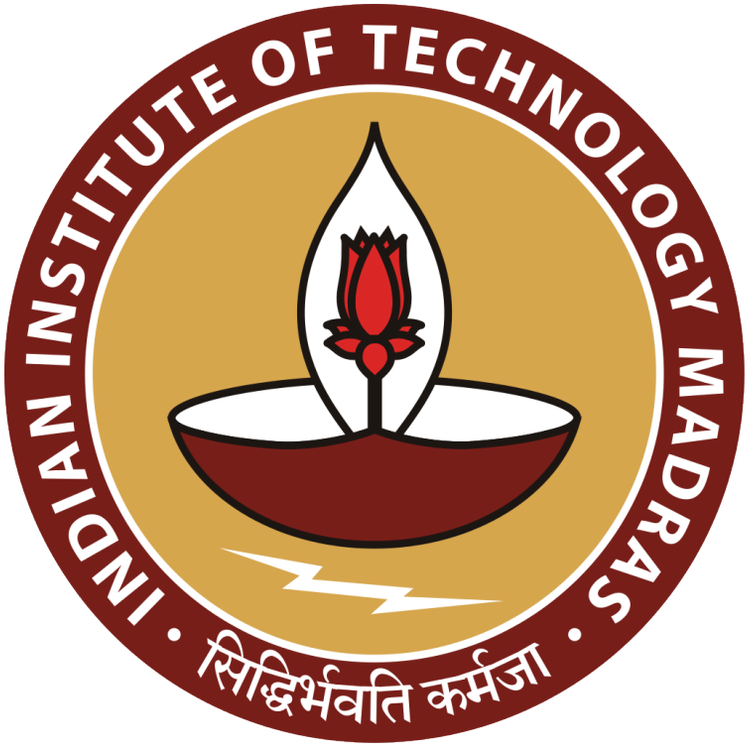

IIT Madras

ONLINE DEGREE

Pseudocode: Side effects

A procedure to sum up any type of marks

```
Procedure SumMarks(gen,fld)
    Sum = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      Move X to Pile 2
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    return(Sum)
end SumMarks
```

A procedure to sum up any type of marks

- Two parameters, gender (**gen**) and field (**fld**)

Procedure SumMarks(gen,fld)

```
Sum = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
return(Sum)
```

end SumMarks

A procedure to sum up any type of marks

- Two parameters, gender (**gen**) and field (**fld**)
- What about the set of cards?

Procedure SumMarks(gen,fld)

```
Sum = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
return(Sum)
```

end SumMarks

A procedure to sum up any type of marks

- Two parameters, gender (**gen**) and field (**fld**)
- What about the set of cards?
- This procedure works for a fixed set of cards

```
Procedure SumMarks(gen,fld)
    Sum = 0
    while (Pile 1 has more cards) {
      Pick a card X from Pile 1
      Move X to Pile 2
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    return(Sum)
end SumMarks
```

A procedure to sum up any type of marks

- Two parameters, gender (**gen**) and field (**fld**)
- What about the set of cards?
- This procedure works for a fixed set of cards
- Pass the deck of cards through a third parameter!

Procedure SumMarks(gen,fld)

```
Sum = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
return(Sum)
```

end SumMarks

Sum up any type of marks for any deck of cards

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    return(Sum)
end SumMarks
```

Sum up any type of marks for any deck of cards

- Third parameter **Deck**

```
Procedure SumMarks(gen,fld, Deck )
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    return(Sum)
end SumMarks
```

Sum up any type of marks for any deck of cards

- Third parameter **Deck**
- New variable **SeenDeck** for second pile

Procedure SumMarks(gen,fld, Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
return(Sum)
```

end SumMarks

Sum up any type of marks for any deck of cards

- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    return(Sum)
end SumMarks
```

Sum up any type of marks for any deck of cards

- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes
  - Cards move from **Deck** to **SeenDeck**

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
return(Sum)
```

end SumMarks

Sum up any type of marks for any deck of cards

- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes
  - Cards move from **Deck** to **SeenDeck**
- At the end of the procedure, **Deck** is empty!

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
return(Sum)
```

end SumMarks

Sum up any type of marks for any deck of cards

- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes
  - Cards move from **Deck** to **SeenDeck**
- At the end of the procedure, **Deck** is empty!
- Procedure should also restore **Deck**

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Sum up any type of marks for any deck of cards

- Third parameter **Deck**
- New variable **SeenDeck** for second pile
- **Deck** is modified as the procedure executes
  - Cards move from **Deck** to **SeenDeck**
- At the end of the procedure, **Deck** is empty!
- Procedure should also restore **Deck**
- Is this sufficient?

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects

- What is the status of **Deck** after the procedure?

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not
  - Depends what we mean by "restore" **Deck**

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not
  - Depends what we mean by "restore" **Deck**
  - **SeenDeck** would normally be in reverse order

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects

- What is the status of **Deck** after the procedure?
- Is each card the same as it was before?
  - We certainly expect so
- Is the sequence of cards the same as it was before?
  - Perhaps not
  - Depends what we mean by "restore" **Deck**
  - **SeenDeck** would normally be in reverse order
- Side effect Procedure modifies some data during its computation

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged
- Sometimes the side effect is the end goal

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged
- Sometimes the side effect is the end goal
  - Procedure to arrange cards in decreasing order of Total Marks

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged
- Sometimes the side effect is the end goal
  - Procedure to arrange cards in decreasing order of Total Marks
- A side effect could be undesirable

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged
- Sometimes the side effect is the end goal
  - Procedure to arrange cards in decreasing order of Total Marks
- A side effect could be undesirable
  - We pass a deck arranged in decreasing order of Total Marks

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

- Sequence of cards may be disturbed
- Does it matter?
  - Not in this case — adding marks does not depend on how the cards are arranged
- Sometimes the side effect is the end goal
  - Procedure to arrange cards in decreasing order of Total Marks
- A side effect could be undesirable
  - We pass a deck arranged in decreasing order of Total Marks
  - After the procedure, the deck is randomly rearranged

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Interface vs implementation

Each procedure comes with a contract

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Interface vs implementation

Each procedure comes with a contract

- Functionality

```
Procedure SumMarks(gen,fld,Deck)
    Sum = 0
    while (Deck has more cards) {
      Pick a card X from Deck
      Move X to SeenDeck
      if (X.Gender == gen) {
        Sum = Sum + X.fld
      }
    }
    Restore Deck from SeenDeck
    return(Sum)
end SumMarks
```

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return
- Data integrity

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return
- Data integrity
  - Can the procedure have side effects?

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return
- Data integrity
  - Can the procedure have side effects?
  - Is the nature of the side effect predictable?

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return
- Data integrity
  - Can the procedure have side effects?
  - Is the nature of the side effect predictable?
    - For instance, deck is reversed

Procedure SumMarks(gen,fld,Deck)

Sum = 0

while (Deck has more cards) {

Pick a card **X** from Deck

Move **X** to SeenDeck

if (**X**.Gender == gen) {

Sum = Sum + **X**.fld

}

}

Restore Deck from SeenDeck

return(Sum)

end SumMarks

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return
- Data integrity
  - Can the procedure have side effects?
  - Is the nature of the side effect predictable?
    - For instance, deck is reversed

Contract specifies interface

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Interface vs implementation

Each procedure comes with a contract

- Functionality
  - What parameters will be passed
  - What is expected in return
- Data integrity
  - Can the procedure have side effects?
  - Is the nature of the side effect predictable?
    - For instance, deck is reversed

Contract specifies interface

- Can change procedure implementation (code) provided interface is unaffected

Procedure SumMarks(gen,fld,Deck)

```
Sum = 0
while (Deck has more cards) {
  Pick a card X from Deck
  Move X to SeenDeck
  if (X.Gender == gen) {
    Sum = Sum + X.fld
  }
}
Restore Deck from SeenDeck
return(Sum)
```

end SumMarks

Side effects . . .

Associating personal pronouns with nouns

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Side effects . . .

Associating personal pronouns with nouns

- Fix a pronoun

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Side effects . . .

Associating personal pronouns with nouns

- Fix a pronoun
- Search text before pronoun backwards

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Side effects . . .

Associating personal pronouns with nouns

- Fix a pronoun
- Search text before pronoun backwards
- Stop at the nearest name

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Side effects . . .

Associating personal pronouns with nouns

- Fix a pronoun
- Search text before pronoun backwards
- Stop at the nearest name

Write a procedure for pronoun matching

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Side effects . . .

Associating personal pronouns with nouns

- Fix a pronoun
- Search text before pronoun backwards
- Stop at the nearest name

Write a procedure for pronoun matching

- **FindMatch(before, pronoun, after)**
  - Three parameters

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Side effects . . .

Associating personal pronouns with nouns

- Fix a pronoun
- Search text before pronoun backwards
- Stop at the nearest name

Write a procedure for pronoun matching

- **FindMatch(before, pronoun, after)**
  - Three parameters
- **FindMatch** should not disturb **before** and **after**
  - Sequence of words, position

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Side effects . . .

Associating personal pronouns with nouns

- Fix a pronoun
- Search text before pronoun backwards
- Stop at the nearest name

Write a procedure for pronoun matching

- **FindMatch(before, pronoun, after)**
  - Three parameters
- **FindMatch** should not disturb **before** and **after**
  - Sequence of words, position
- No side effects should happen

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head-Master with his thin long cane.

Summary

- Need to separate interface and implementation

Summary

- Need to separate interface and implementation
- Interface describes a contract

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible
- Can change the implementation provided we preserve the interface

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible
- Can change the implementation provided we preserve the interface
- Side effects are important to be aware of

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible
- Can change the implementation provided we preserve the interface
- Side effects are important to be aware of
  - Sometimes no guarantee is needed (adding up mark)

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible
- Can change the implementation provided we preserve the interface
- Side effects are important to be aware of
  - Sometimes no guarantee is needed (adding up mark)
  - Sometimes no side effect is tolerated (pronoun matching)

Summary

- Need to separate interface and implementation
- Interface describes a contract
  - Parameters to be passed
  - Value to be returned
  - What side effects are possible
- Can change the implementation provided we preserve the interface
- Side effects are important to be aware of
  - Sometimes no guarantee is needed (adding up mark)
  - Sometimes no side effect is tolerated (pronoun matching)
  - Sometimes the side effect is the goal (sort the data)