Today

Multiple Inheritance

- Composite figures through multiple inheritance
- Multiple inheritance: Combining abstractions
- Multiple inheritance: Name clashes
- Results of renaming
Acceptable name clashes

If inherited features have all the same names, there is no harmful name clash if:

- They all have compatible signatures
- At most one of them is effective

Semantics of such a case:
- Merge all features into one
- If there is an effective feature, it imposes its implementation

Feature merging

Feature merging: with different names

Feature merging: effective features

Repeated inheritance

Assume class TAXPAYER with attributes

- age: INTEGER
- address: STRING
- bank_account: ACCOUNT
- tax_id: INTEGER

and routines such as

- pass_birthday is
  do
  age := age + 1
  end
- pay_taxes is ...
- deposit_to_account (sum: INTEGER) is ...

Heirs may include SWISS_TAXPAYER and US_TAXPAYER.
Repeated inheritance

The two above classes may in turn have a common heir: SWISS_US_TAXPAYER.

Repeated inheritance issues

What happens with features inherited twice from the common ancestor TAXPAYER, such as address, age, tax_id, pass_birthday?

Sharing and replication

Features such as age and birthday, not renamed along any of the inheritance paths, will be shared. Features such as tax_id, inherited under different names, will be replicated.

The inheritance clause

class SWISS_US_TAXPAYER
inherit
  SWISS_TAXPAYER
rename
    address as swiss_address,
    tax_id as swiss_tax_id,
    pay_taxes as pay_swiss_taxes,
    ...end...
US_TAXPAYER
rename
    address as us_address,
    tax_id as us_tax_id,
    pay_taxes as pay_us_taxes,
    ...end...
end

The need for select

Assume there is a redefinition somewhere along the way:

A potential ambiguity arises because of polymorphism and dynamic binding:

  t: TAXPAYER
  su: SWISS_US_TAXPAYER

  t := su
  print t.address
Removing the ambiguity

```java
class SWISS_US_TAXPAYER
  inherit SWISS_TAXPAYER
  rename
    address as swiss_address,
    tax_id as swiss_tax_id,
    pay_taxes as pay_swiss_taxes,
    bank_account as swiss_bank_account,
    deposit_to_account as deposit_to_swiss_account
  select
    swiss_address,
    swiss_tax_id,
    pay_swiss_taxes,
    swiss_bank_account,
    deposit_to_swiss_account
  end
end
```

When is a name clash acceptable?

(Between n features of a class, all with the same name, immediate or inherited.)

- They must all have compatible signatures.
- If more than one is effective, they must all come from a common ancestor feature under repeated inheritance.

```java
class US_TAXPAYER
  rename
    address as us_address,
    tax_id as us_tax_id,
  select
    us_address,
    us_tax_id
  end
```

---

Intro. to Programming, lecture 16: Multiple inheritance & project presentation

19

Removing the ambiguity

```java
class SWISS_US_TAXPAYER
  inherit SWISS_TAXPAYER
  rename
    address as swiss_address,
    tax_id as swiss_tax_id,
    pay_taxes as pay_swiss_taxes,
    bank_account as swiss_bank_account,
    deposit_to_account as deposit_to_swiss_account
  select
    swiss_address,
    swiss_tax_id,
    pay_swiss_taxes,
    swiss_bank_account,
    deposit_to_swiss_account
  end
end
```

When is a name clash acceptable?

(Between n features of a class, all with the same name, immediate or inherited.)

- They must all have compatible signatures.
- If more than one is effective, they must all come from a common ancestor feature under repeated inheritance.

```java
class US_TAXPAYER
  rename
    address as us_address,
    tax_id as us_tax_id,
  select
    us_address,
    us_tax_id
  end
```