ソフトウェア開発のSOFL形式工学手法の紹介と課題

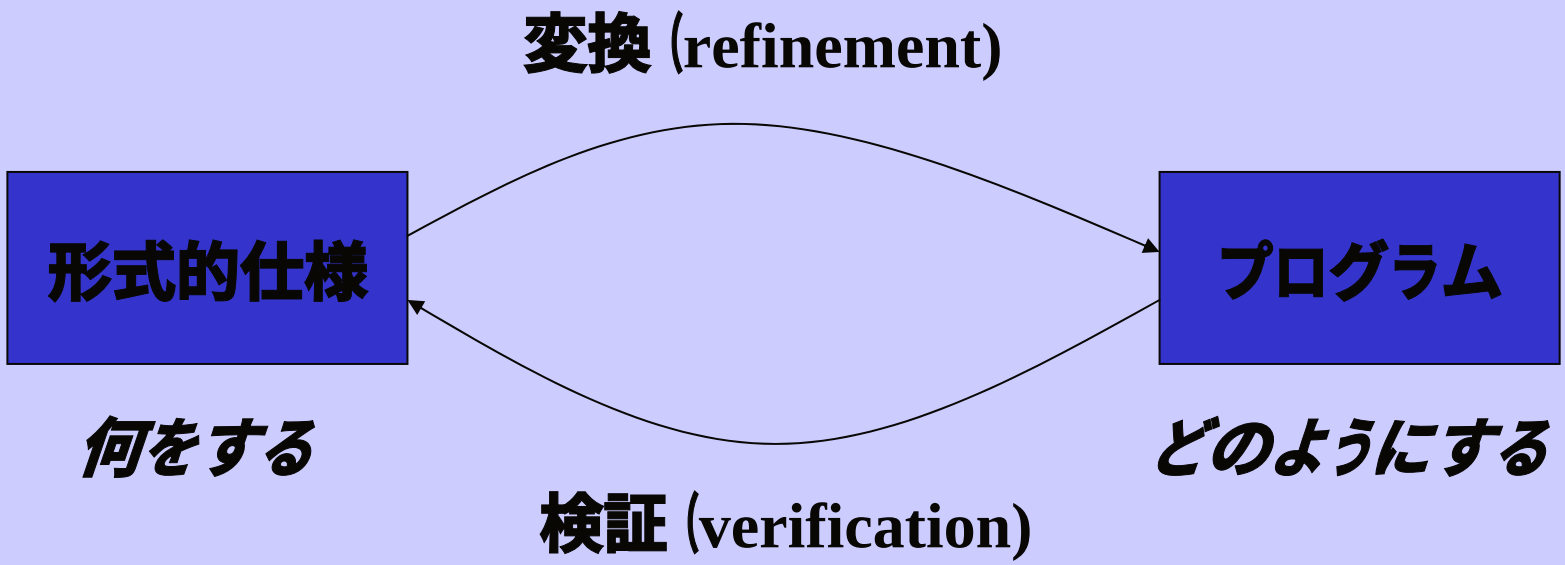
(Shaoying Liu) **劉 少英**

法政大学 情報科学部

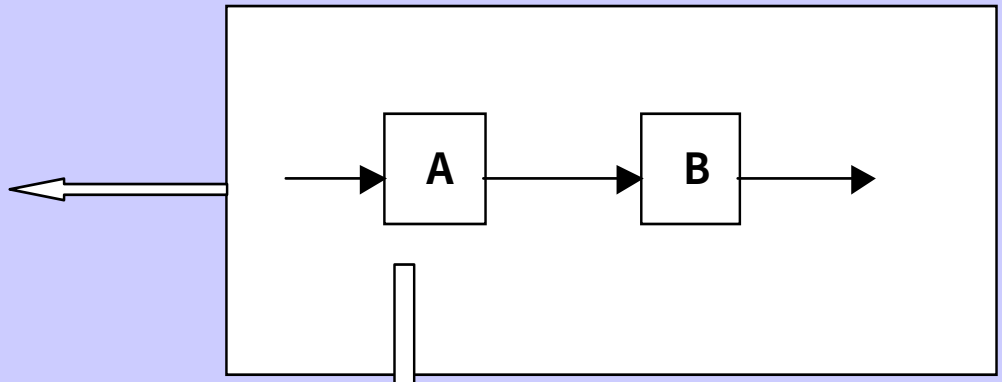
Email:sliu@k.hosei.ac.jp

URL: http://www.k.hosei.ac.jp/~sliu/

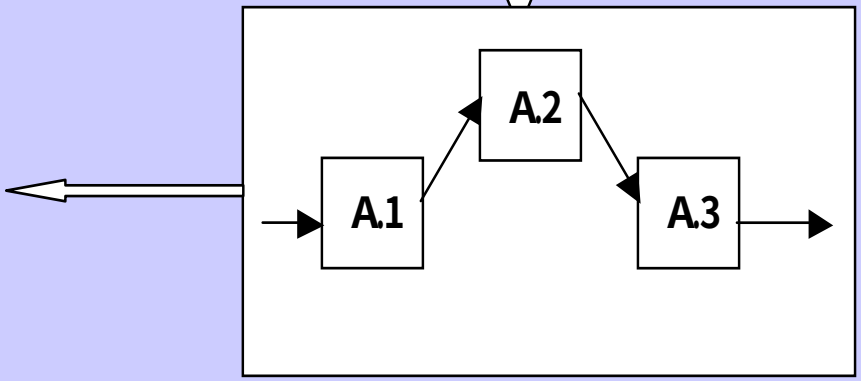
形式工学手法


変換 (refinement)
形式的仕様
プログラム
何をする
どのようにする
検証 (verification)
支える
図、離散数学 (集合論、論理) 、自然言語

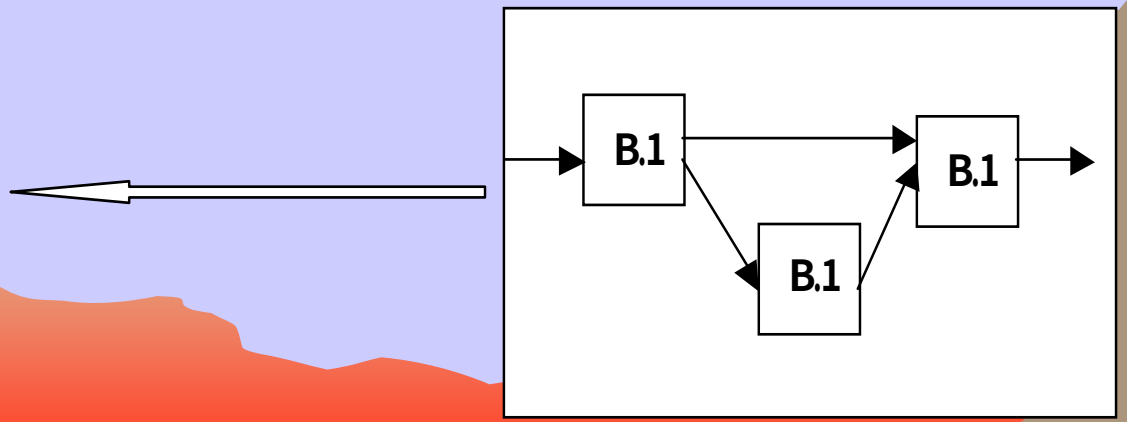
```
module System-Example;
process A;
process B;
end-module;
```




```
module A-Decom;
process
process A.2
process A.3;
end-module;
```



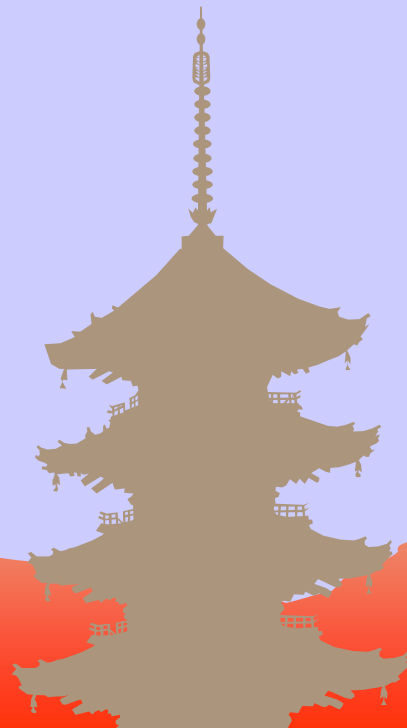

```
module B-Decom;
process B.1;
process B.2;
process B.3;
end-module;
```



Process specification

The general structure of a process specification:

process A(input variables) output variables
ext external variable
pre P
post Q
explicit
  variable declarations
  statement
comment
end_process;



課題1

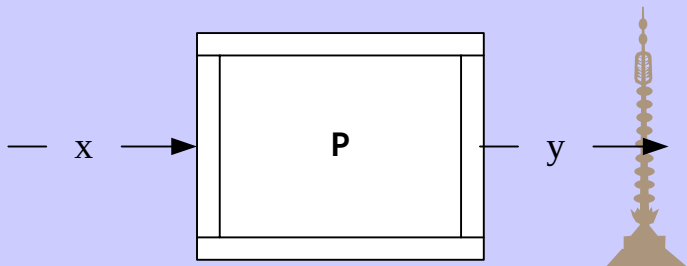
次のプロセス仕様をJavaで実装しなさい。

process P(x: int) y: int

pre x > 0

post y > x + 1

end_process



課題2

次のプロセス仕様をJavaで実装しなさい。

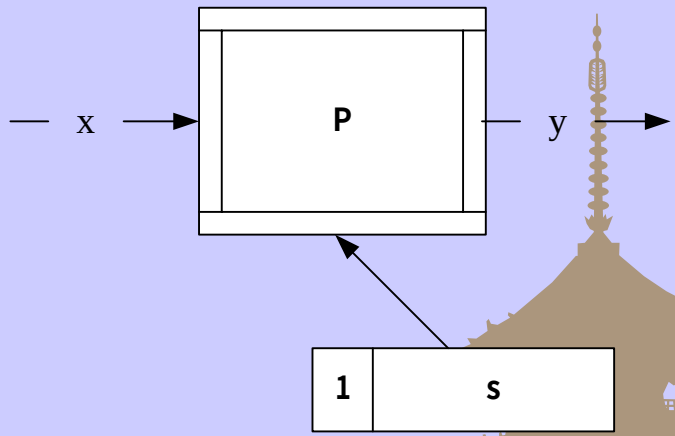
process P(x: int) y: int

ext rd s: int

pre x > s

post y > s and y < x

end_process



課題3

次のプロセス仕様をJavaで実装しなさい。

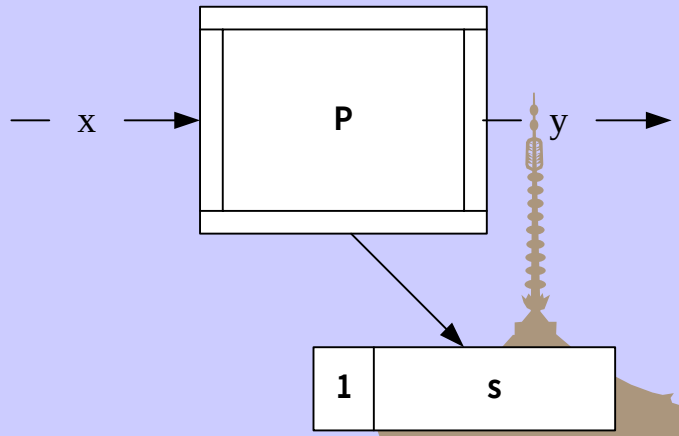
process P(x: int) y: int

ext wr s: int

pre x > s

post s > ~s + x and s = y + x

end_process



課題4

次のプロセス仕様をJavaで実装しなさい。

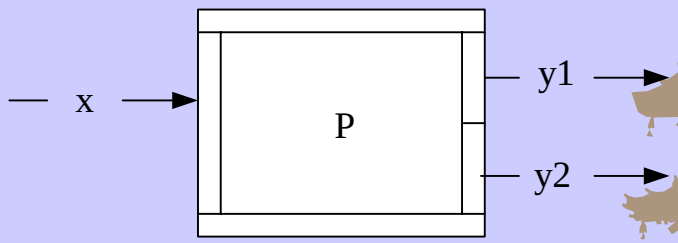
process P(x: int) y1: int | y2: real

pre x > 0

post x < 5 and y1 = x + 5 or

x >= 5 and y2 = (x * x + 2 * x + 5) / x

end_process



課題5

次のプロセス仕様をJavaで実装しなさい。

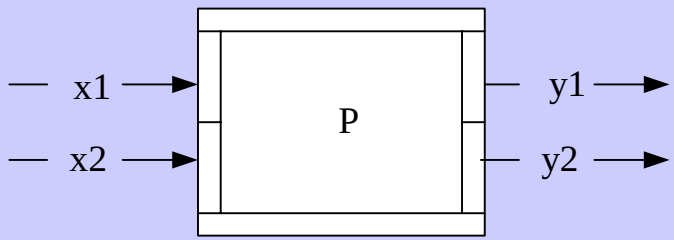
process P(x1: int | x2: int) y1: int | y2: real

pre x1 > 0 or x2 > 0

post bound(x1) and y1 > x1 + 5 or

bound(x2) and y2 = (x * x + 2 * x + 5)

end_process

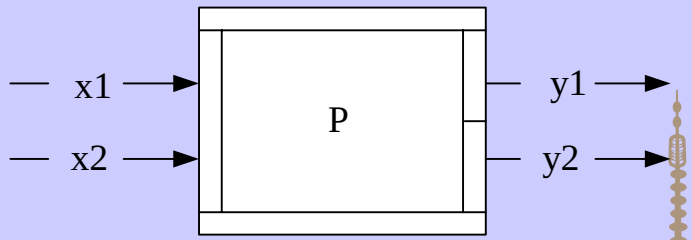


課題6

次のプロセス仕様をJavaで実装しなさい。

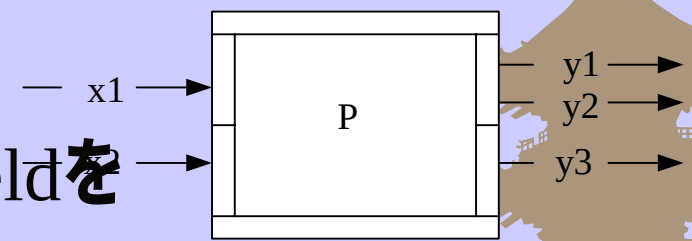
(1)

```
process P(x1, x2: int) y1: int | y2: real
pre x1 <> 0
post if x1 > x2
        then y1 = x1 * x2
        else  y2 = x2 / x1
end_process
```



(2)

```
process P(x1: int | x2: int) y1, y2: int | y3: int
 pre x1 > 5 or x2 >= 3
 post if x1 < 10
        then y1 > x1 + 1 and y1 < y2 – x1
        else  if x2 < 20
                then y3 > x2 + 5 or y3 < x2 + 1
                else y3 = -10
 end_process
```



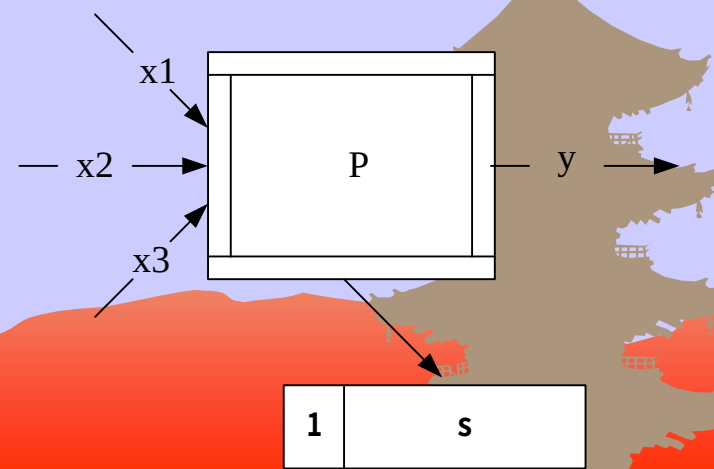
(注意：無定義の場合。TextFieldを使う)

課題7

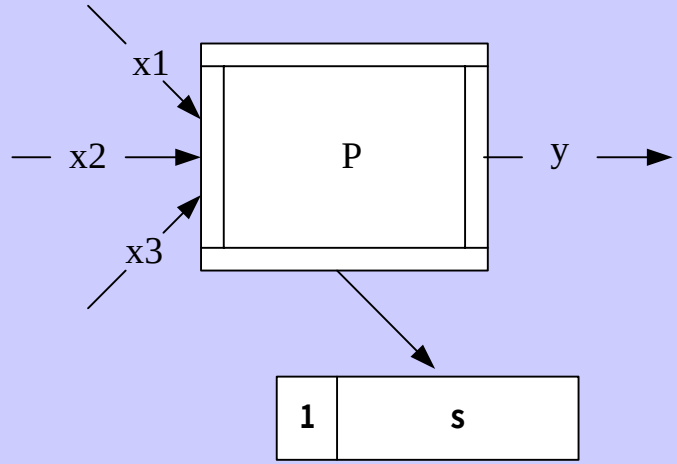
次のプロセス仕様をJavaで実装しなさい。

(1)

```
process P(x1, x2, x3: int) y: int
ext wr s: int
post  let d = x1 ** 3 + x2 * x3 in
        if x1 > x2 + x3
        then y = d ** 2 + 5 * d and s >  ~s + 1
        else  y > ~s + d and s = ~s * d
end_process
```



(2)



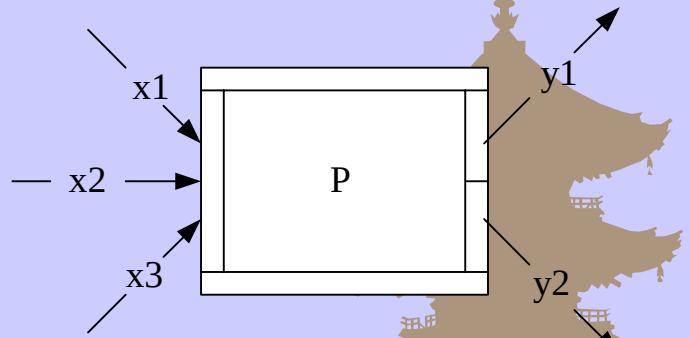

```
process P(x1, x2, x3: int) y: int
ext wr s: int
post  let d = x1 ** 3 + x2 * x3 in
        if x1 > x2 + x3
        then cases x1
                x2 + x3 + 1→ y = d ** 2 + 5 * d and
                                 s >  ~s + 1
                x2 + x3 + 2 →y > ~s + d and s = ~s * d
                x2 + x3 + 5 →y = d ** 2 and s >  ~s + d
                others → y = d
        else  y > d + x1 * x2 * x3
end_process
```

課題8

次のプロセス仕様をJavaで実装しなさい。

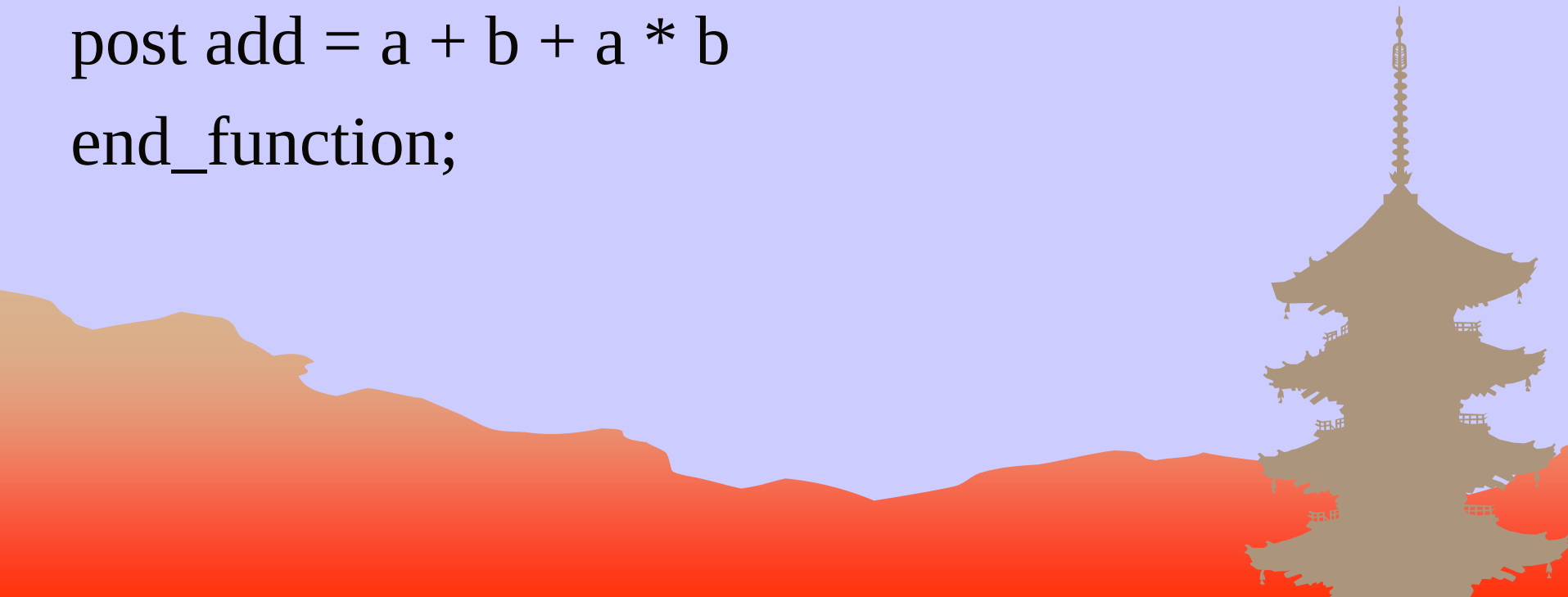
(1)

```
process P(x1, x2, x3: int) y1: int | y2: int
pre is_greater(x1, x2)
post is_greater(x1, x3) and y1 = x1 * double(x3, x2) or
        is_greater(x3, x2) and y2 = x2 * add(x3, x1)
end_process;
function is_greater(a, b: int): bool
== a > b
end_function
```

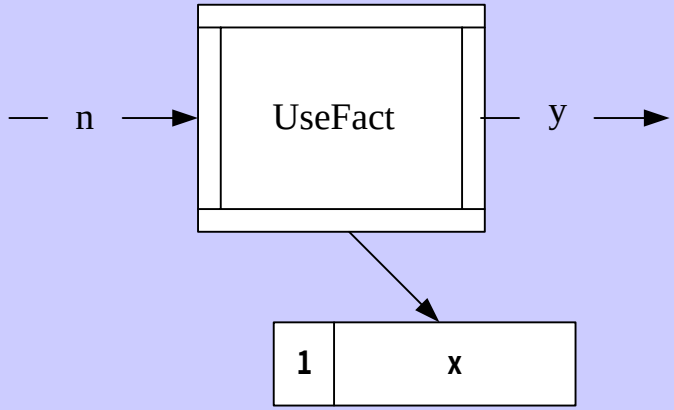


```
function double(a, b: int): int
== 2 * (a + b)
end_function;

function add(a, b: int): int
post add = a + b + a * b
end_function;
```



(2)



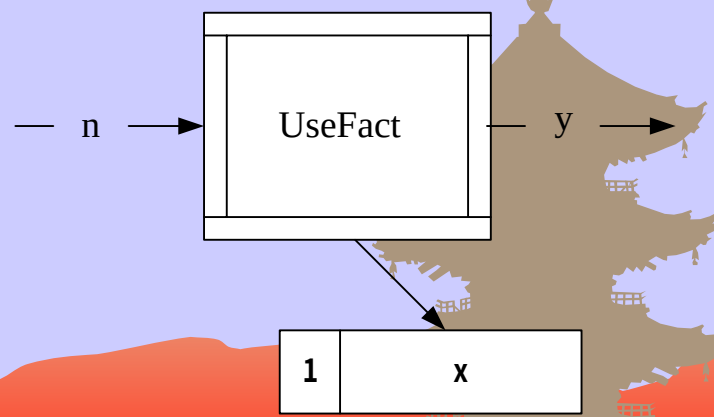

```
process UseFact(n: nat) y: nat
 ext wr x: nat
 pre n > x
 post y – (Fact(n) + Fact(~x)) * (Fact(n – 1) + Fact(~x + 1))
        = 0 and x = double(~x, add(~x, ~x + 1))
 end_process;
 function Fact(n: nat): nat
 Fact(n) == if n = 1
                then 1
                else n * Fact(n – 1)
end_function
```

課題9

次のプロセス仕様をJavaで実装しなさい。

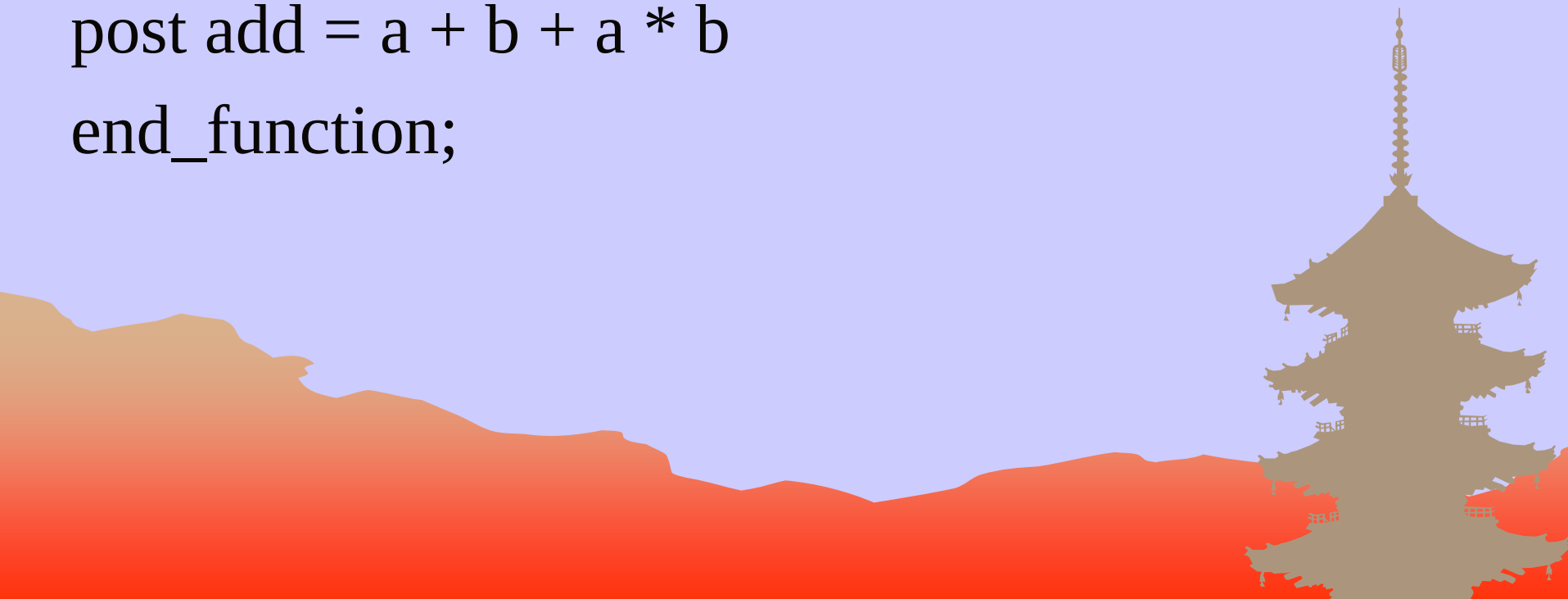
(1)

```
process UseFact(n: nat) y: nat
ext wr x: nat
pre n > x
post y – (Fact(2 * n) + Fact(~x)) * (Fact(n) + Fact(~x + 1))
        = 0 and x = double(~x, add(~x, ~x + 1))
end_process;
function Fact(n: nat): nat
Fact(n) == if n = 1
                then 1
                else n * Fact(n – 1)
end_function
```



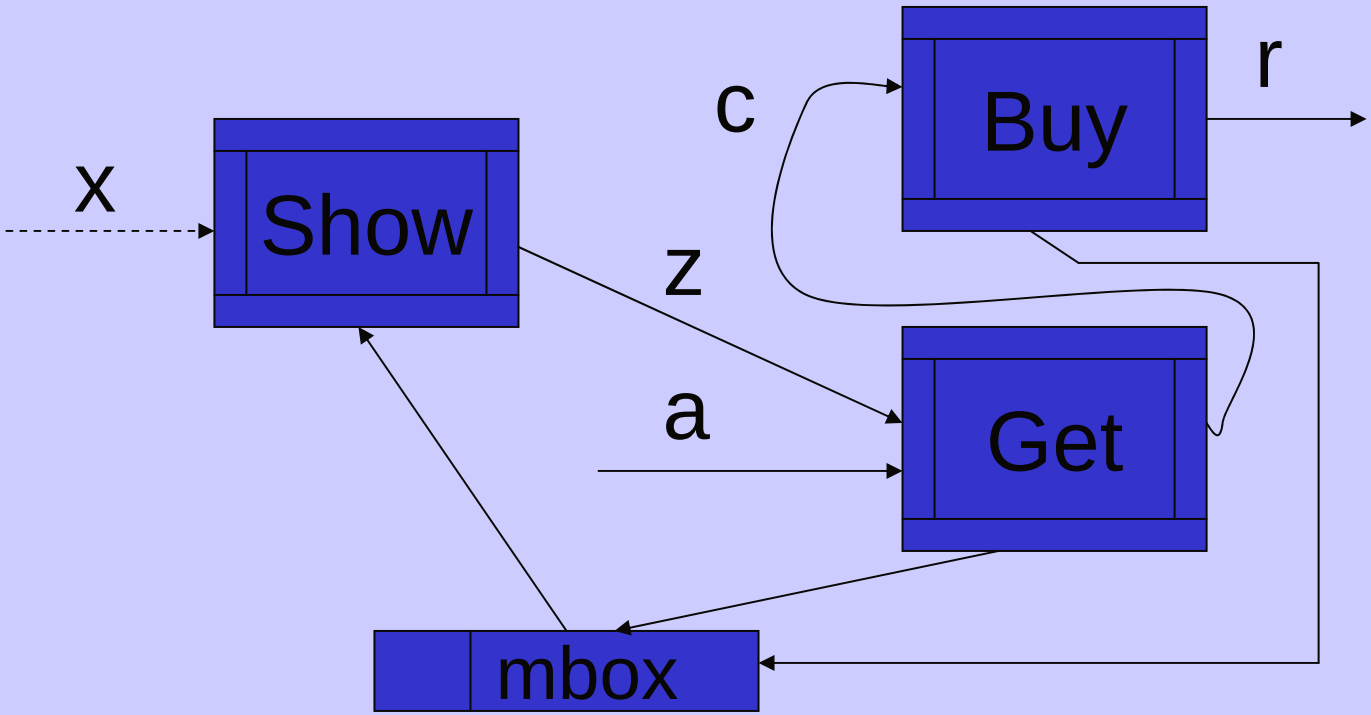
```
function double(a, b: int): int
 == 2 * (a + b)
 end_function;

 function add(a, b: int): int
 post add = a + b + a * b
 end_function;
```



課題１０

次のCDFDをJavaで実装しなさい。



mbox: moneybox

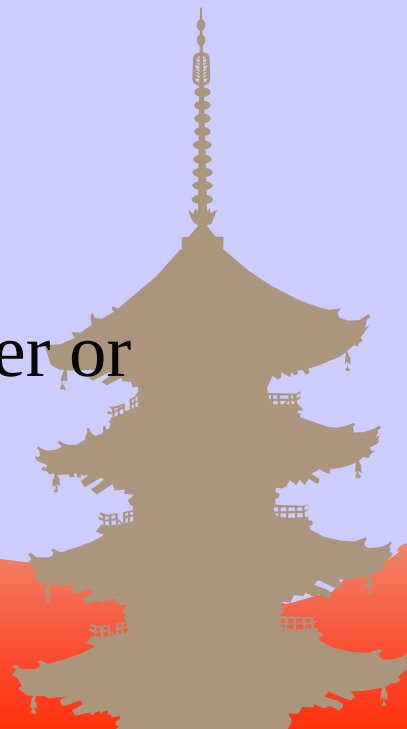
Figure 1

```
module System_Moneybox
const toyprice = 98;  /* The price of a specific toy */
var
mbox: nat;
behav CDFD_Figure 1;
process Show(x : Sign) z : nat
ext rd mbox  /* rd = read */
pre true       /* This precondition can be omitted */
post z = mbox
comment
```

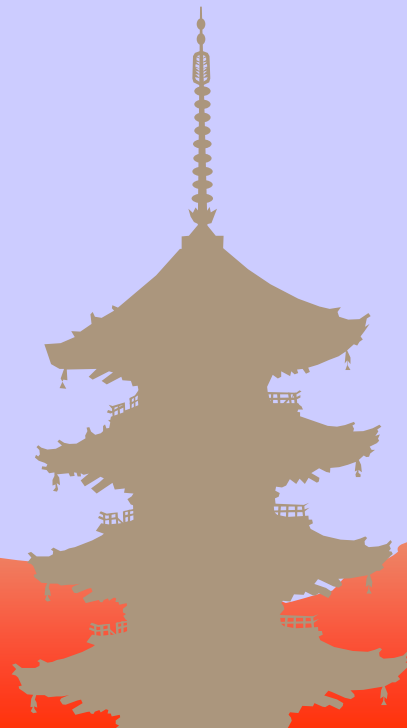
Show is executed on the availablility of event x, under no specific condition, and will generate the output z that is equal to the money in the money box mbox.

```
end_process;
```

```
process Get(z : nat, a : nat) c : nat
ext wr mbox : nat  /*wr = write */
                   /* : nat can be omitted */
pre    z >= a
post   c >= a and mbox = z – c
comment
 If input z, the total money in the mbox, is not
 smaller than a, an estimated amount of money
 necessary to buy a toy, the output
 c, the money taken from the mbox, must be greater or
 equal to a, and mbox must be updated with c.
end_process;
```

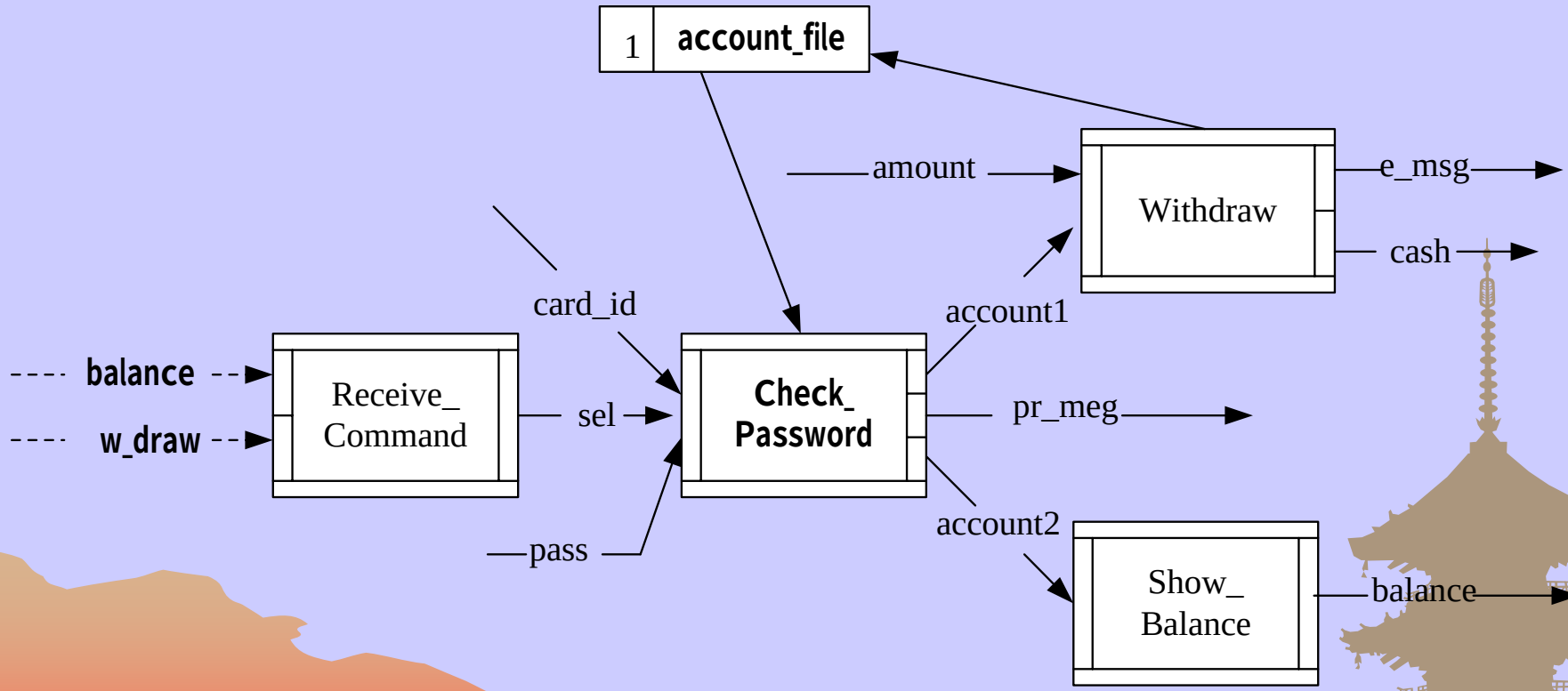


```
process Buy(c : nat) r : nat
ext wr mbox
pre c >= toyprice
post r = c – toyprice and mbox = ~mbox + r
comment
 If money c is greater than or equal
 to the toyprice, output r will represents
 the remaining money after buying the toy
 with the toyprice, and put the remaining
 money r back to the money box, which
 is expressed by the expression:
 mbox = ~mbox + r.
end_process;
end_module;
```



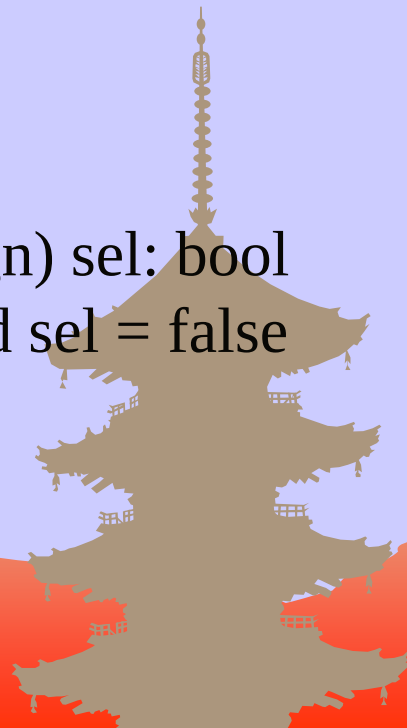
課題１１

次のCDFDをJavaで実装しなさい。



```
module SYSTEM_ATM;
  type
        Account = composed of
          account_no: nat0
          password: PassWord
          balance: real
        end
PassWord = nat0;
  var
  ext #account_file: set of Account;

 process Receive_Command(balance: sign | w_draw: sign) sel: bool
  post balance <> nil and sel = true or w_draw <> nil and sel = false
  comment
  If the input is balance,   set sel as true;     otherwise,
  set sel as false.
  end_process;
```



```
process Check_Password(card_id: nat, sel: bool, pass: nat)
                account1: Account | pr_meg: string |
                account2: Account
ext rd account_file
post  sel = false and
      (exists![x: account_file] | x.account_no = card_id and
                                  x.password = pass and
                                   account1 = x)
        or
       sel = true and
       (exists![x: account_file] | x.account_no = card_id and
                                   x.password = pass and
                                   account2 = x)
        or
       not (exists![x: account_file] | x.account_no = card_id and
                                       x.password = pass) and
                                pr_meg = "Reenter your password or insert the correct card"
```

comment

If there exists an account, say x, in the account_file whose account number is the same as card_i and password is the same as pass, then (1) if sel is false, then assign x to account1 as an output, (2) if sel is true, then assign x to account2 as an output; otherwise, issue an appropriate message to indicate the absence of the account.

end_process;

```
process Withdraw(amount: real, account1: Account)
                        e_msg: string | cash: real
ext wr account_file
pre account1 inset account_file
post (exists[x: account_file] | x = account1 and
                                  x.balance >= amount and
                                   cash = amount)
      and
      account_file = union(diff(~account_file, {account1}),
 {modify(account1, balance -> account1.balance - amount)})        or
    not exists[x: account_file] | x = account1 and
                                   x.balance >= amount and
                        e_meg = "The amount is too big")
comment
    If the balance of account1 is greater than or equal to amount, then assign amount to cash
     and reduce amount from the balance of account1;
    otherwise, if the amount is greater than the balance of account1, then issue an message to
     indicate the lack of sufficient money to withdraw from account1.    end_process;
end_module;
```