Comp 110
Loops

Instructor: Prasun Dewan

Prerequisite

- Conditionals

.

Looping

printHello(2);

hello

hello

printHello(3);

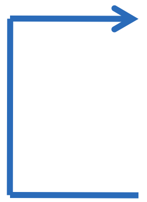
hello

hello

hello

Loops

```
public static void printHellos(int n) {

        int  counter = 0;
        if (counter < n) {
                counter  = counter + 1;
                System.out.println (“hello”);
        }

}
```



Loops

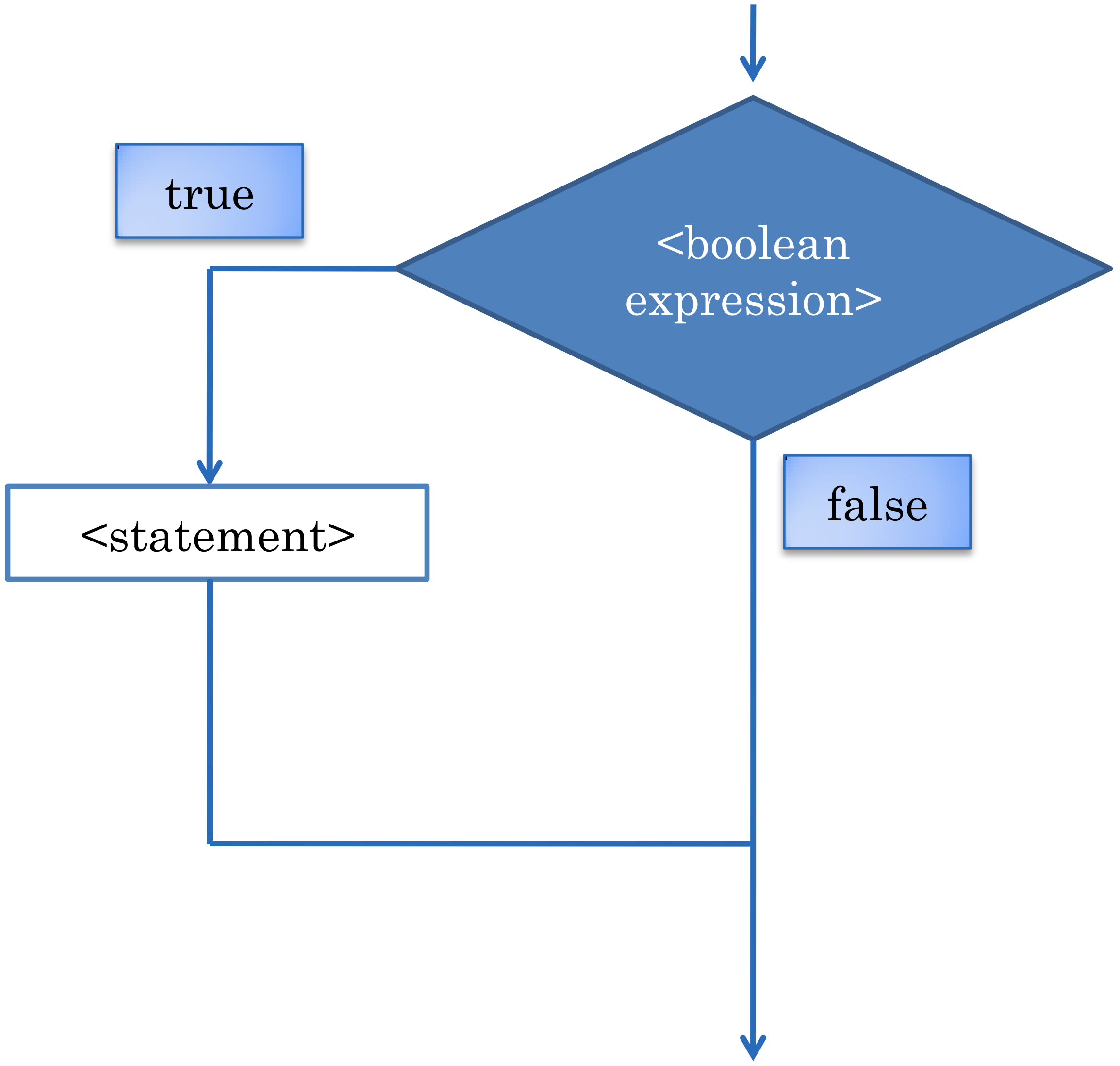
```
public static void printHellos(int n) {

        int  counter = 0;
        while (counter < n) {
                counter  = counter + 1;
                System.out.println ("hello");
        }

}
```

If vs. While Statement

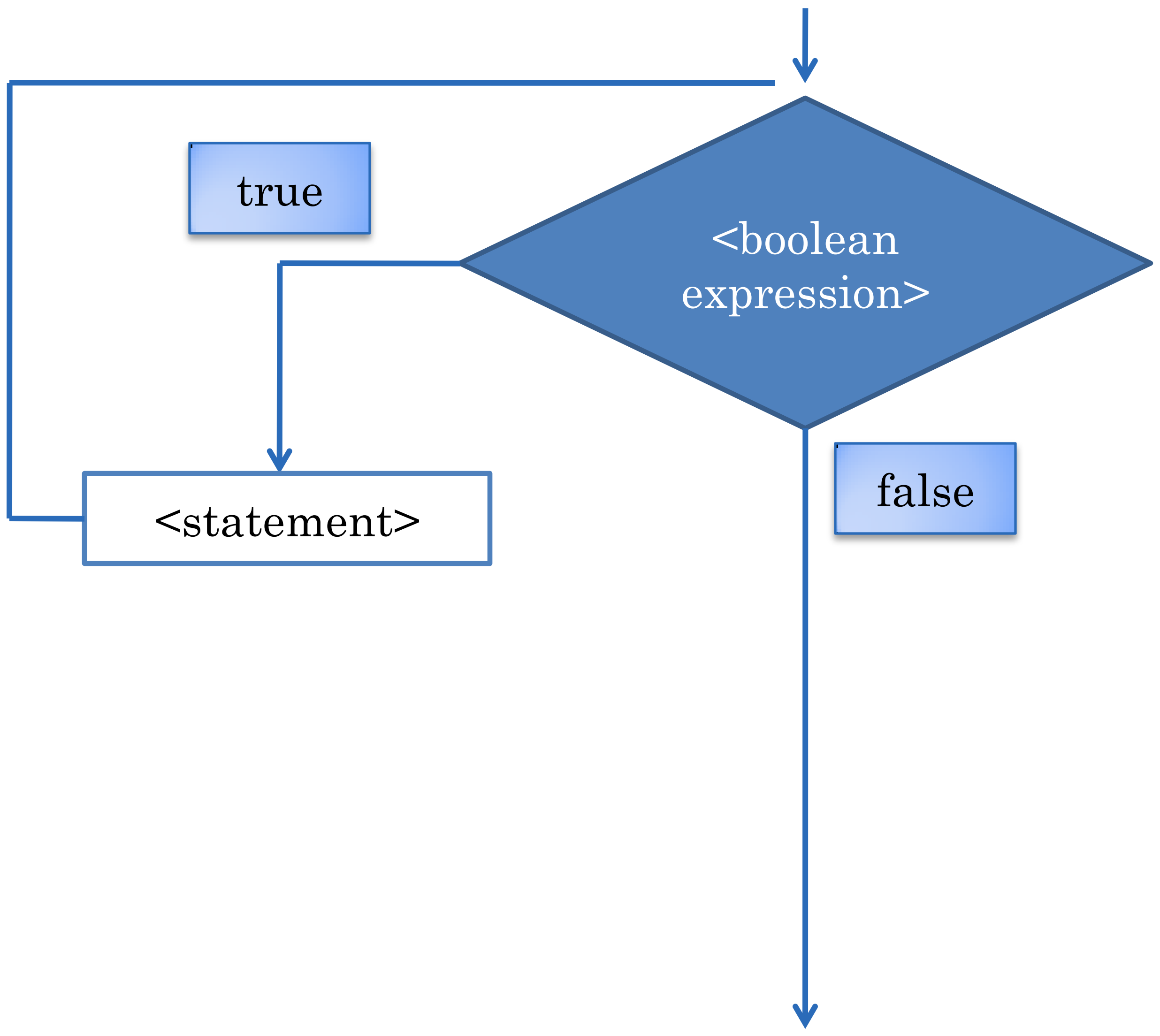
```
if (<bool expr>)
        <statement>;
```

```
while (<bool expr>)
        <statement>;
```

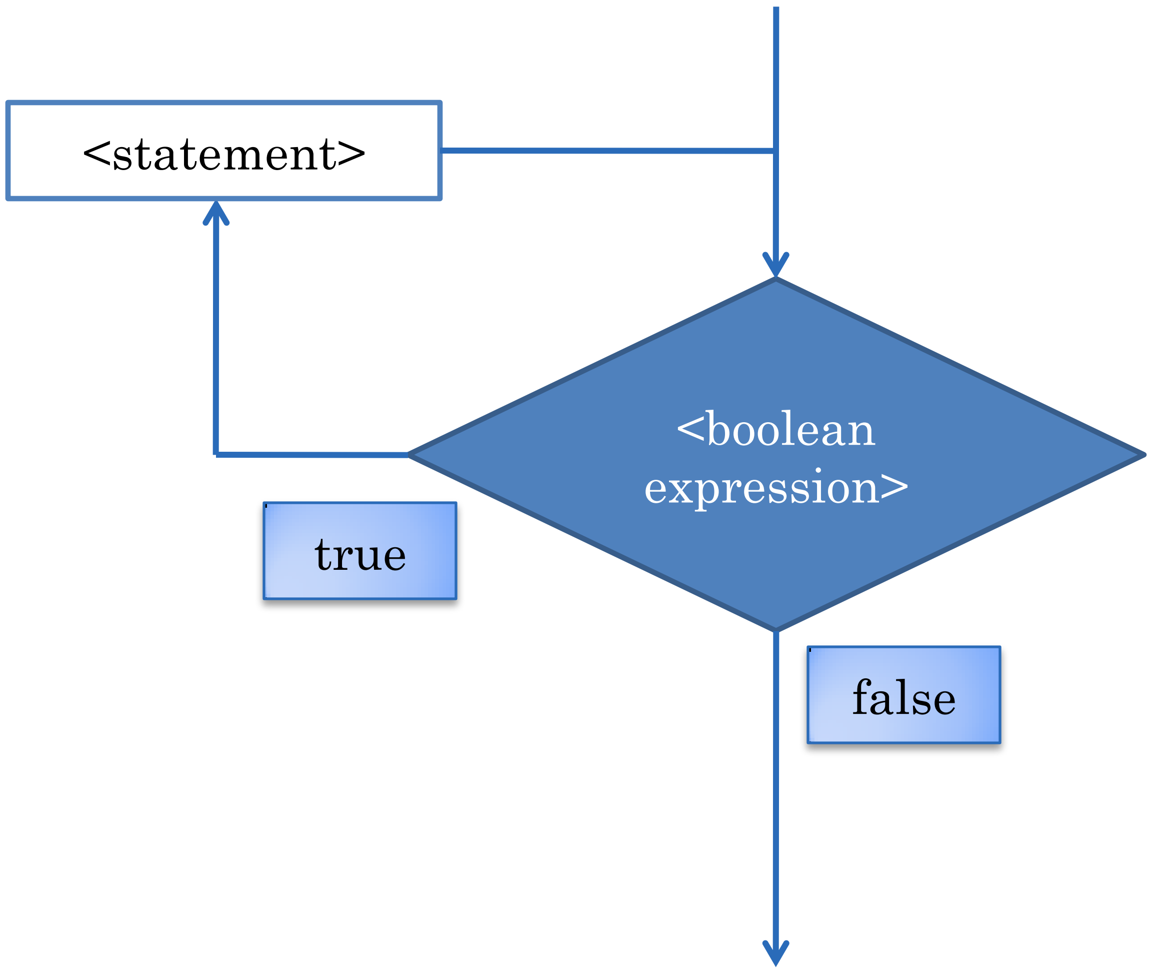
If Statement



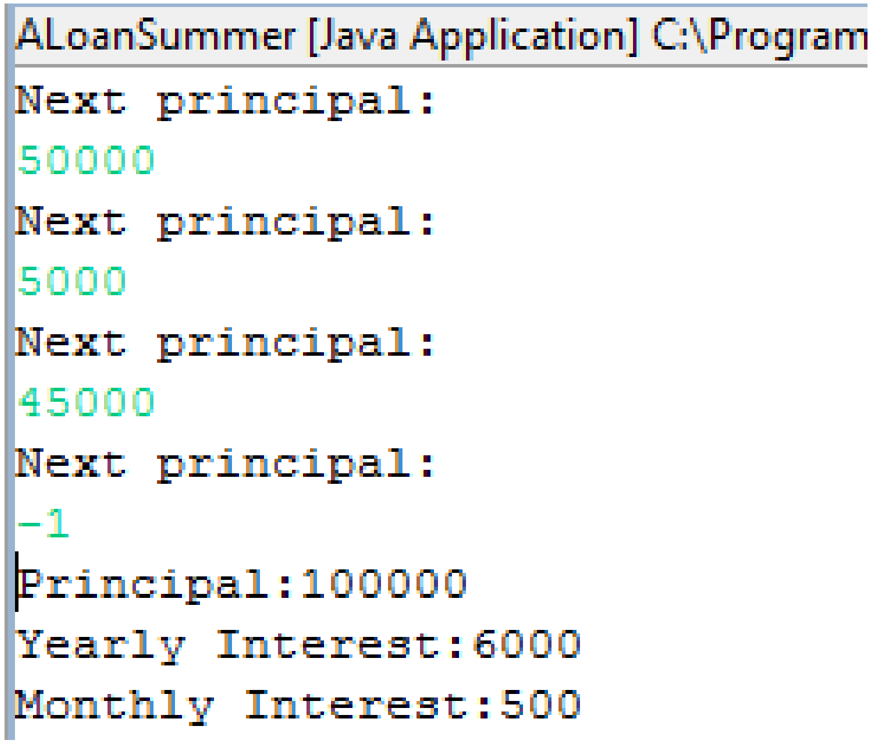
While Statement



WHILE LOOP



Sentinel-Based Folding



ALoanSummer [Java Application] C:\Program

```
Next principal:
50000
Next principal:
5000
Next principal:
45000
Next principal:
-1
Principal:100000
Yearly Interest:6000
Monthly Interest:500
```

Adding Fixed Number of Loans

```
Loan loan1 = readLoan();
Loan loan2 = readLoan();
Loan loan3 = readLoan();
Loan loan4 = readLoan();
Loan sumLoan = ALoan.add( loan1,
            ALoan.add(loan2,
                        ALoan.add(loan3, loan4))
            );
print(sumLoan);
```

Generalizing to Variable Number of Loans

```
Loan loan1 = readLoan();
Loan loan2 = readLoan();
Loan loan3 = readLoan();
Loan loan4 = readLoan();
…
Loan loanN = readLoan();
```

Variable Number of Statements

Loops and Arrays

```
Loan sumLoan = ALoan.add(loan1,
           ALoan.add(loan2,
                      ALoan.add(loan3,
                                 ALoan.add(loan4, ……(
                                            ALoan.add(loanN-1, loanN)*;
```

```
print (sumLoan);
```

Variable Number of Subexpressions (function calls)

Recursion

Space-Efficient Adding of Fixed Number of Loans

```
Loan loan1 = readLoan();
Loan loan2 = readLoan();
Loan sumLoan = ALoan.add(loan1, loan2);
loan1 = readLoan(); // 3rd loan
sumLoan = ALoan.add(sumLoan, loan1);
loan1 = readLoan(); // 4th loan
sumLoan = ALoan.add(sumLoan, loan1);
print (sumLoan);
```

More Space-Efficient Adding of Fixed Number of Loans

```
Loan sumLoan = readLoan(); //first loan
Loan nextLoan = readLoan(); //second loan
sumLoan = ALoan.add(nextLoan, sumLoan);
nextLoan = readLoan(); // 3rd loan
sumLoan = ALoan.add(sumLoan, nextLoan);
nextLoan = readLoan(); // 4th loan
sumLoan = ALoan.add(sumLoan, nextLoan);
print (sumLoan);
```

More Space-Efficient Adding of Variable Number of Loans

```
Loan sumLoan = readLoan(); //first loan
Loan nextLoan = readLoan(); //second loan
sumLoan = ALoan.add(nextLoan, sumLoan);
nextLoan = readLoan(); // 3rd loan
sumLoan = ALoan.add(sumLoan, nextLoan);
nextLoan = readLoan(); // 4th loan

sumLoan = ALoan.add(sumLoan, nextLoan);
nextLoan = readLoan(); //Nth loan
sumLoan = ALoan.add(sumLoan, nextLoan);
nextLoan = readLoan(); //sentinel
print (sumLoan);
```

N-1 Repetitions

WHILE LOOP

```
Loan sumLoan = readLoan(); //first loan
          Loan nextLoan = readLoan(); //second loan
          while (nextLoan().getPrincipal() >= 0) {
                    sumLoan = ALoan.add(nextLoan, sumLoan);
                    nextLoan = readLoan(); // next loan or sentinel
          }
          print (sumLoan);
```

Input	Result
-1	Program waits forever for second loan

Boundary Condition

Correct Solution

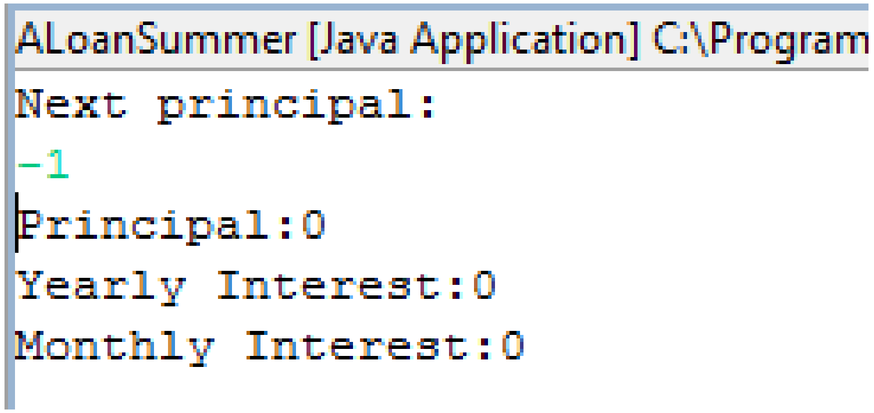
```
Loan sumLoan = new ALoan(0); //initial value
Loan nextLoan = readLoan(); //second loan
while (nextLoan().getPrincipal() >= 0) {
        sumLoan = ALoan.add(nextLoan, sumLoan);
        nextLoan = readLoan(); // next loan or sentinel
}
print (sumLoan);
```

```
ALoan.add(new ALoan(0), add(loan1, add (…., loanN)
```

Identity

A Single Sentinel Value

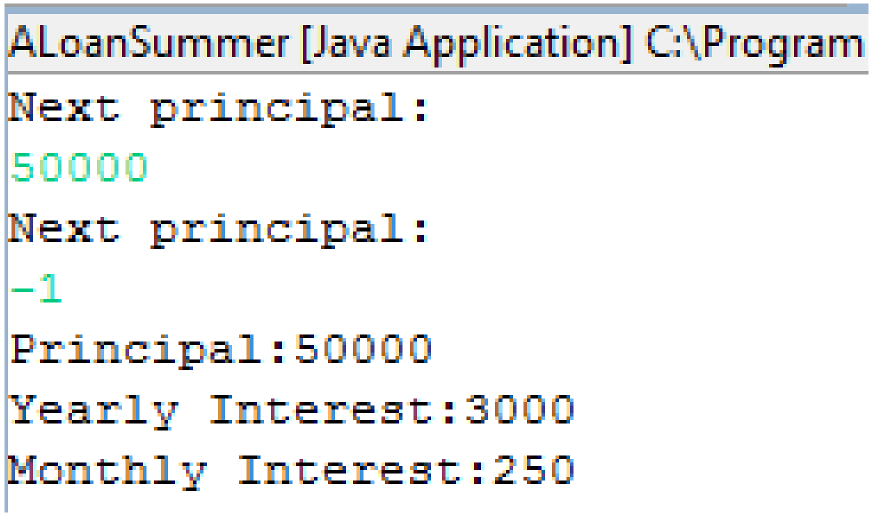
```
Loan sumLoan = new ALoan(0); //initial value
Loan nextLoan = readLoan(); //second loan
while (nextLoan().getPrincipal() >= 0) {
        sumLoan = ALoan.add(nextLoan, sumLoan);
        nextLoan = readLoan(); // next loan or sentinel
}
print (sumLoan);
```



```
ALoanSummer [Java Application] C:\Program
Next principal:
-1
Principal:0
Yearly Interest:0
Monthly Interest:0
```

A Single Loan

```
Loan sumLoan = new ALoan(0); //initial value
Loan nextLoan = readLoan(); //second loan
while (nextLoan().getPrincipal() >= 0) {
        sumLoan = ALoan.add(nextLoan, sumLoan);
        nextLoan = readLoan(); // next loan or sentinel
}
print (sumLoan);
```

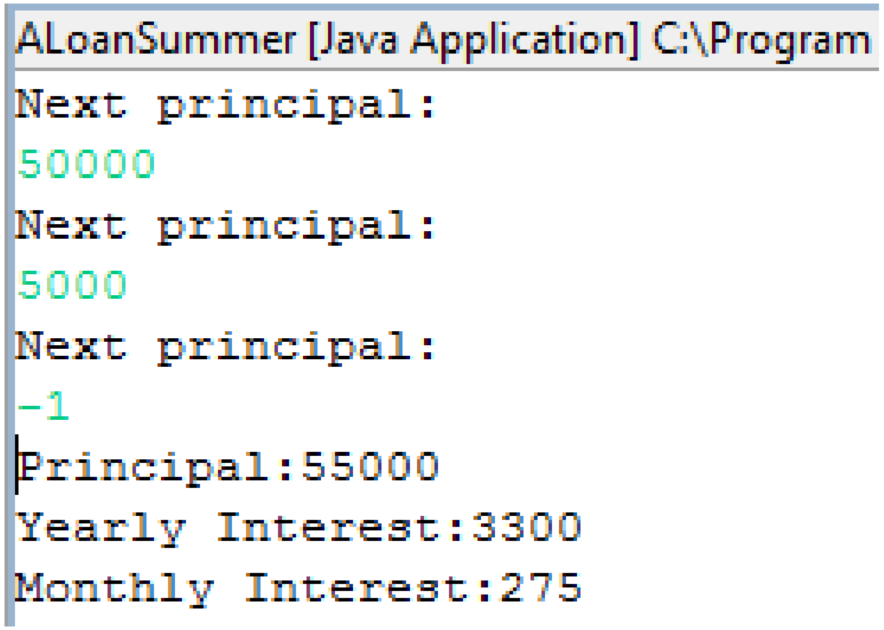


ALoanSummer [Java Application] C:\Program

```
Next principal:
50000
Next principal:
-1
Principal:50000
Yearly Interest:3000
Monthly Interest:250
```

Two Loans

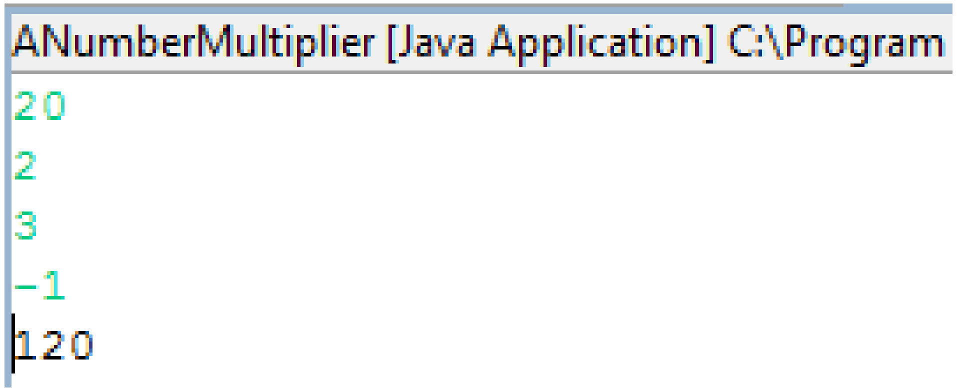
```
Loan sumLoan = new ALoan(0); //initial value
Loan nextLoan = readLoan(); //second loan
while (nextLoan().getPrincipal() >= 0) {
        sumLoan = ALoan.add(nextLoan, sumLoan);
        nextLoan = readLoan(); // next loan or sentinel
}
print (sumLoan);
```



```
ALoanSummer [Java Application] C:\Program
Next principal:
50000
Next principal:
5000
Next principal:
-1
Principal:55000
Yearly Interest:3300
Monthly Interest:275
```

Multiplying Numbers (Edit)

```
public class ANumberMultiplier {
        public static void main(String[] args) {
                int product = 1;
                int nextInt = Console.readInt();
                while (nextInt >= 0) {
                        product = product * nextInt;
                        nextInt = Console.readInt();
                }
                System.out.println(product);
        }
}
```



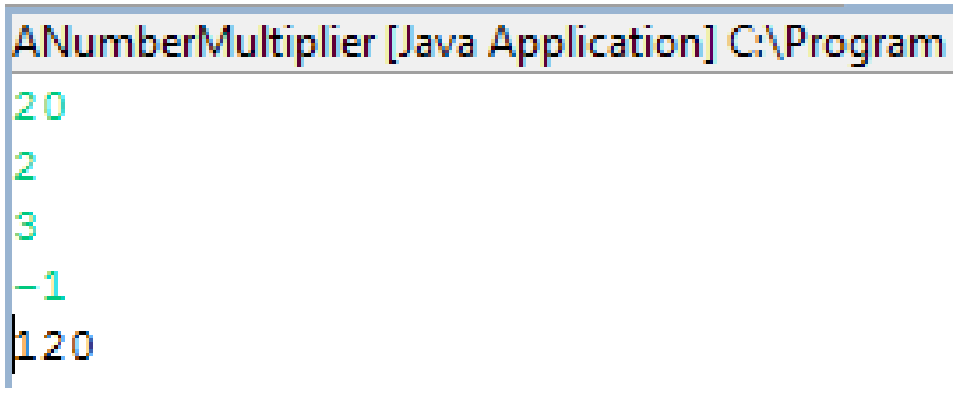
```
ANumberMultiplier [Java Application] C:\Program
20
2
3
-1
120
```

MULTIPLYING NUMBERS

```
int product = 1;
int num = Console.readInt();
while (num >= 0) {
        product = product*num;
        num = Console.readInt();
}
print (product);
```

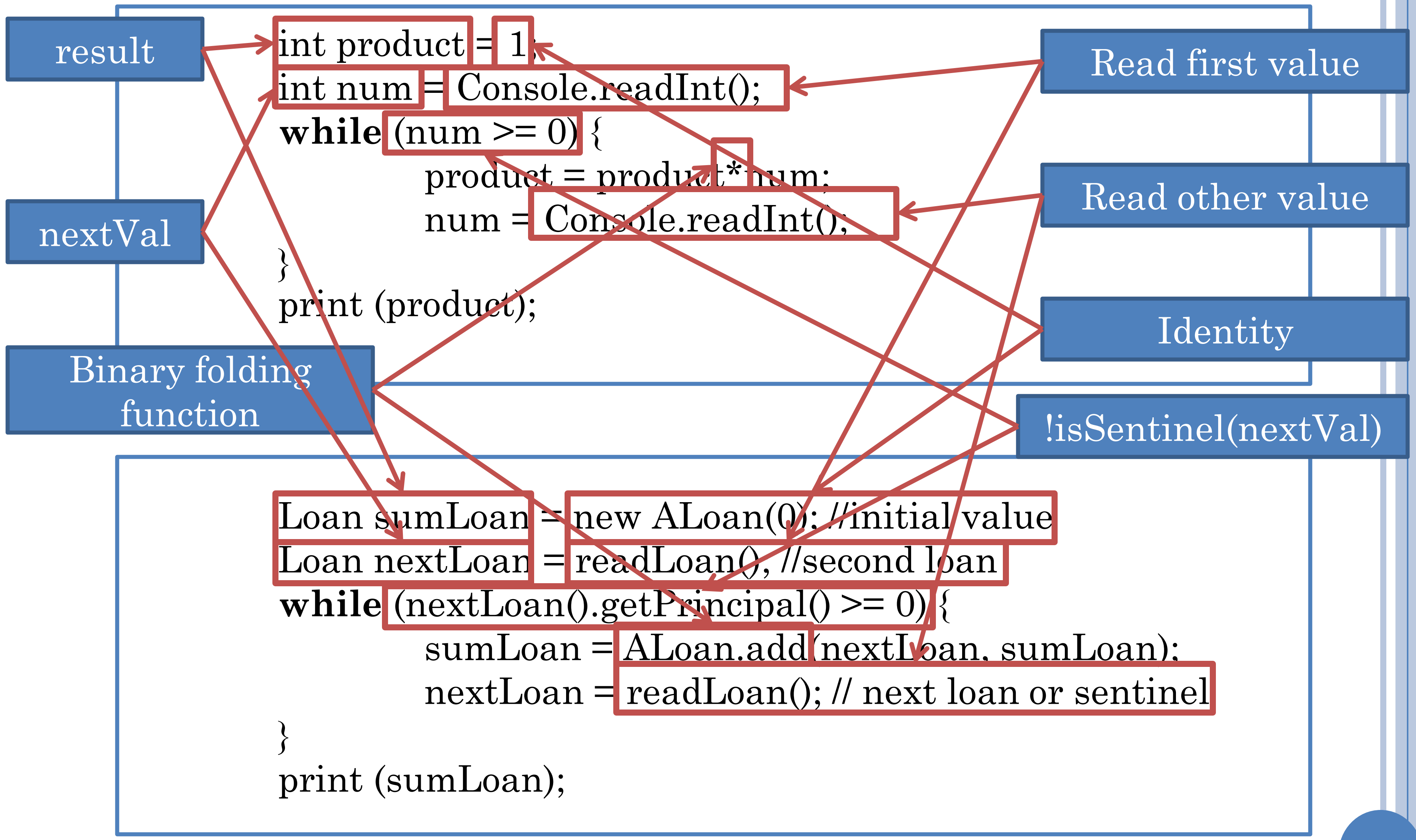
1 * 20 * 2 * 3

Identify



```
ANumberMultiplier [Java Application] C:\Program
20
2
3
-1
120
```

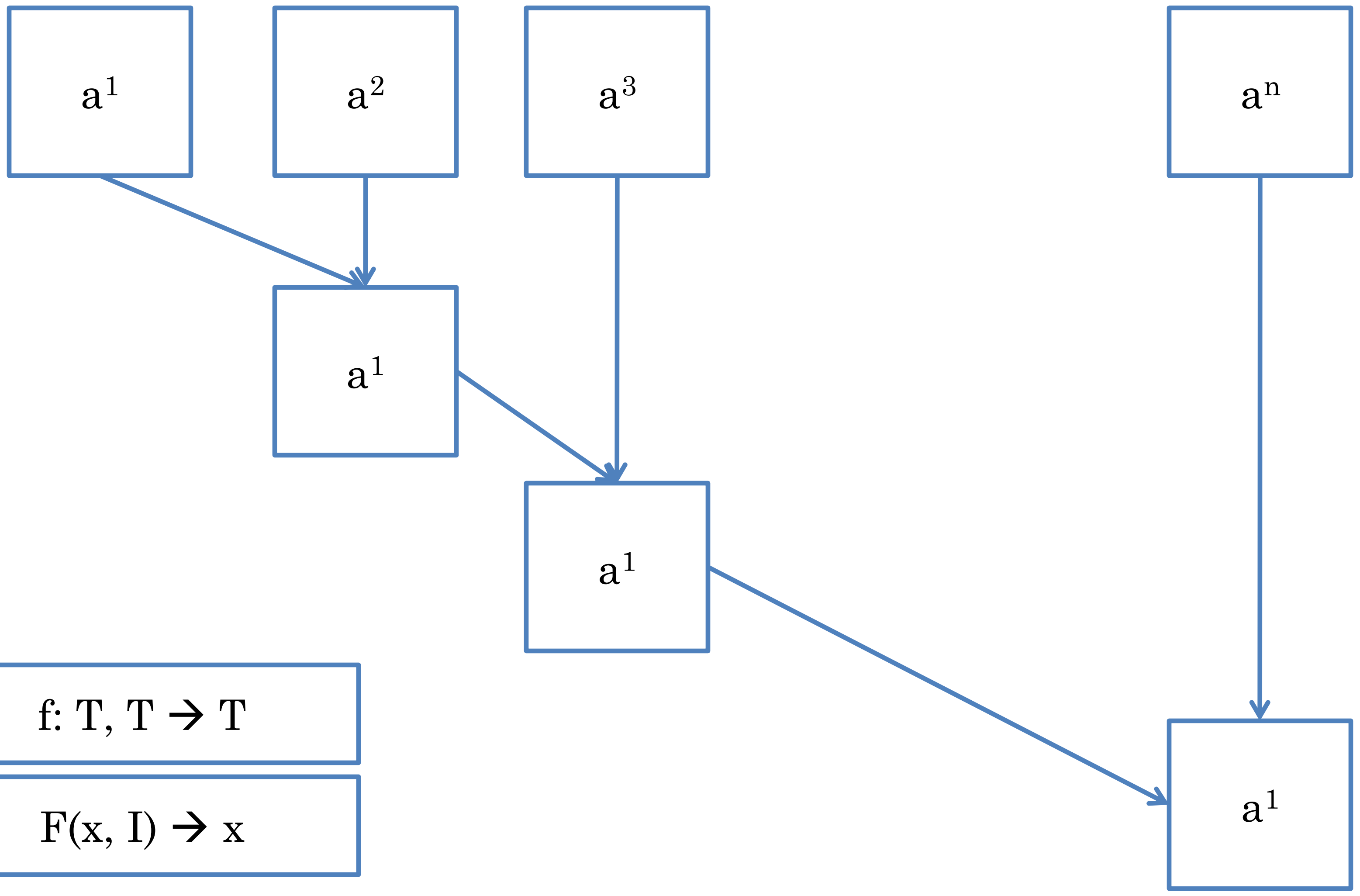
Comparing Two Solutions



```
int product = 1;
int num = Console.readInt();
while (num >= 0) {
        product = product* num;
        num = Console.readInt();
}
print (product);
```

```
Loan sumLoan = new ALoan(0); //initial value
Loan nextLoan = readLoan(); //second loan
while (nextLoan().getPrincipal() >= 0) {
        sumLoan = ALoan.add(nextLoan, sumLoan);
        nextLoan = readLoan(); // next loan or sentinel
}
print (sumLoan);
```

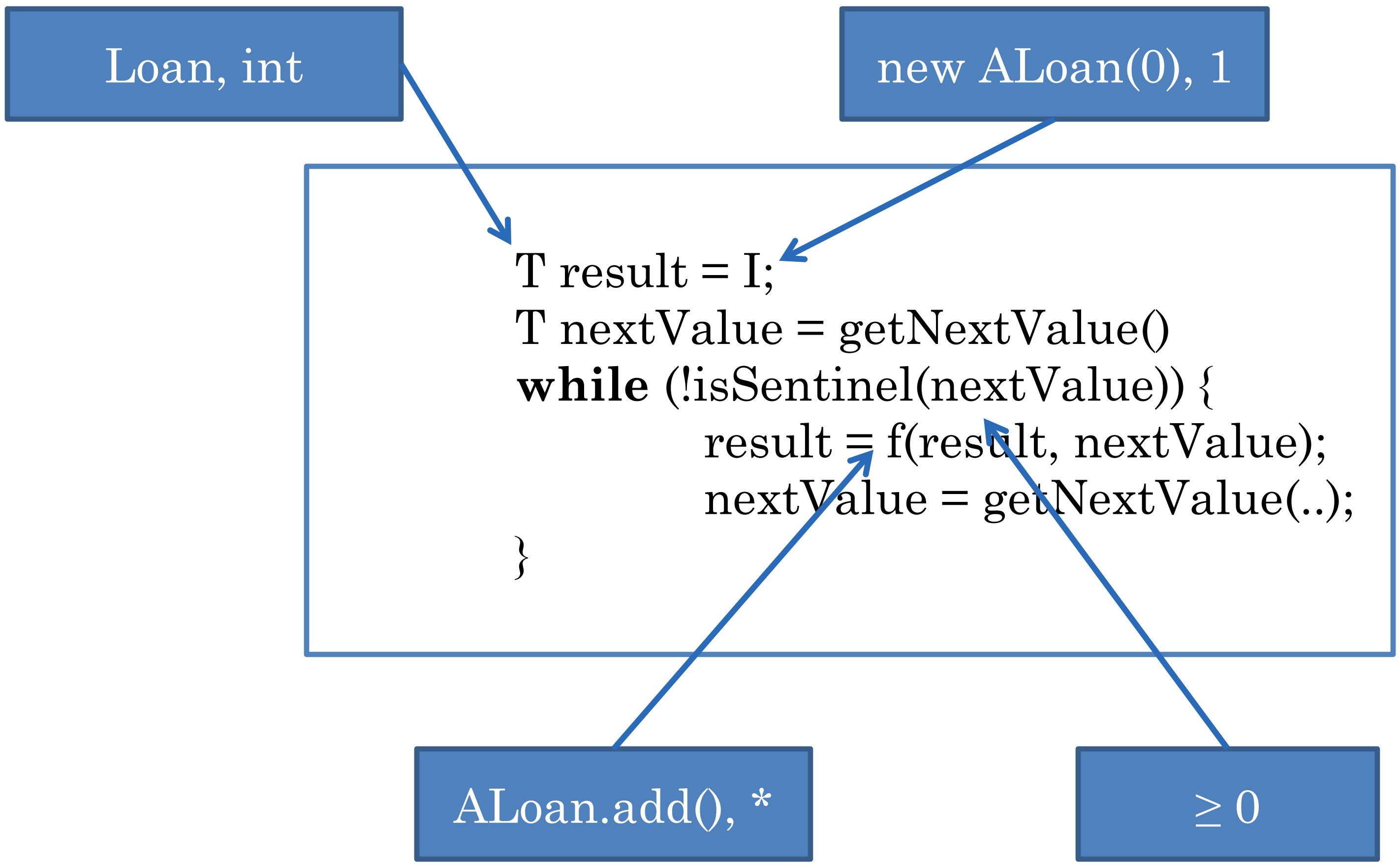
Generalized Folding of a Sentinel-Terminated List



f: T, T → T

F(x, I) → x

Generalized Folding Function



```
T result = I;
T nextValue = getNextValue()
while (!isSentinel(nextValue)) {
            result = f(result, nextValue);
            nextValue = getNextValue(..);
}
```

Comparing Two Solutions (Comments)

```
int product = 1; //identity
int num = Console.readInt(); // read next list value
while (num >= 0) { // sentinel checking
        product = product*num; // binary folding function
        num = Console.readInt(); // read next value
}
print (product);// print value
```

```
Loan sumLoan = new ALoan(0); //identity
Loan nextLoan = readLoan(); // read next list value
while (nextLoan().getPrincipal() >= 0) {// sentinel checking
        sumLoan = Aloan.add(nextLoan, sumLoan); // binary folding function
        nextLoan = readLoan(); // read next list value
}
print (sumLoan); // print value
```