In last lecture we discussed mechanisms to break execution from serial execution.
We use following mechanisms to break serial execution.

1. Break (break)

A break statement can occur inside a loop or inside a switch case statement.

Meaning of the statement is that execution of the loop or the switch case statement is over. Control is passed to the next statement outside loop. In case of switch-case statement control is passed next statement after the switch-case block of statements.

Use of break in switch-case statement:

The conditional expression in switch statement only defines the starting point where the execution shall resume after matching the expression.
It does not specify where to stop; hence all statements in all cases thereafter will get executed.

Example 1:

```
switch( month ){
        case 1 :        last = 31 ;

        case 2 :        last = 28 ;

        default : last = 30;
}
```

In above example, even if month is January value of last will be 28. This happens because even though case 1 was matched case 2 also got executed.

Example 2:

```
switch( month ){
        case 1 :        last = 31 ; break;

        case 2 :        last = 28 ; break;

        default : last = 30;
}
```

In above example if month is January value of last will be 31 after switch-case statement. It will be 28 if month is February.

Example 3:

```
switch( month ){
        case 1 :        last = 31 ; break;

        case 2 :        last = 28 ;
                        if( leapyear () )
                        last = 29;
                        break;
        case 3 :
        case 5 :
        case 7 :
        case 8 :
        case 10 :
        case 12 : last = 31 ; break
        default : last = 30; break;
}
```

Above code sets last to 31 if month is 1,3,5,7,8,10 or 12. Last will be set to 28or 29 in February. While for rest of the months value of last will be 30.

Use of break in loop statement:

By using break we can end the execution of loop, and control resumes after the loop.

Example 4:

```
for ( i=1; i<20; i++ ){

        statement 1 ;

        for( j=1; j<20; j++ ){
                statement 2;
                if ( i == j )
                break;                  // break 1
        }

        statement 3 ;
        break;                          //break 2
}
statement 4;
```

In above example when i equal j, break statement in inner loop will get executed and control will be passed to statement 3. When break statement from outer loop is executed, control will be passed to statement 4.

2. Continue

Continue statement can only appear inside a loop.

It means to skip the current iteration of the loop and continue with the next iteration. All the statements after continue until the end of the loop are not executed for the current iteration, instead control is passed to the loop condition statement.

Example 5:

```
while( s.hasNextInt() ){

        t=s.nextInt() ;
        if ( t < 0 )   continue;
        if ( t > 20 )   continue;
        System.out.println(" Fact (" + t + " ) = " + fact (t) );
}
```

In above example if t is less than 0 or greater than 20, println statement won't be executed. Instead we continue with next iteration.
So if *t = -5* or *t = 25 ,* we ignore.

3.Exceptions

System gives us error message in case of error in the execution of program. For ex. When we try to access elements outside the array bound we get an error.

Default behavior in case of error condition is that program gets terminated.
Error conditions are known as *exceptions.* Java has a class Exception. Methods can throw exceptions. Exceptions can be caught using a try and catch block.

```
Try {
        Statement list ;
} catch ( Exception e) {
        print error message ;
}
```

Statement *throw* will throw an exception using an object of class Exception.

Q. Any method to move pointer of scanner go back? Can we scan again, the input which is already scanned?
→ No. We can't read input which is already read.