Control statements

- Selection stmt : to choose between different paths
  - If , switch
- Iteration stmt : to repeat statements
  - For, while, do-while
- Jump stmt : transfer control from one location to another
  - Break, continue, return

- Real world java appl & applets are graphical & window based not console based
- One console input to read one char from KB
- System.in.read()
- Char is returned as int. you need to typecast to char
- Console input is line buffered until you press return key
- Must use throws java.io.Exception clause to handle input errors

```
// Input Statement

class Input2
{

   public static void main(String args[])
   throws java.io.IOException
   {
      char op;
      int x=20, y=50;
      op = (char) System.in.read();
      switch(op)
      {
        case '+' : System.out.print("x+y=" + (x+y) ); break;
          case ‘-' : System.out.print("x-y=" + (x-y) ); break;
           case ‘*' : System.out.print("x*y=" + (x*y) ); break;
           case ‘/' : System.out.print("x/y=" + (x/y) ); break

      }

   }
}
```

If stmt

If (condition) stmt else stmt

Else block is optional

If can be nested / if-else ladder

Else always refer to nearest unpaired if

Switch

- Provide multiway branch
- Through this can be done by nested-if, switch is more efficient approach
- Exp type must be of byte, short, int, char. String is also permitted in new vertions
- Case labels are unique const exp
- Duplicate cases are not allowed
- Default is optional
- Control goes on until break is encountered or default or last case or end of switch
- Break causes control flow to exit from switch
- Can have empty cases
- Nested switch:
  - Case const can be common

```
// Switch statement

class Switch1
{
  public static void main(String args[])
  {
    int i =5;
   for(i=0; i<=9; i++)
      {
        switch(i)
        {
          case 0 : System.out.print("Zero");break;
          case 1:  System.out.print("one"); break;
          case 5 : System.out.print("Five");break;
          case 7 : System.out.print("Seven"); break;
          default : System.out.print(" not 0,1 5");
        }
      }
  }
```

For loop

For(init; condition; iteration) stmt;

- Usually used to repeat predetermined number of times or when sequence of values are required
- Loop can proceed in postive or negative fashion
- Condition is tested at top of loop
- Can have multiple control var & iteration exp seperated by ,
- Init, cond, iteration can be abscent
- Empty cond imply true value ie enter into infinite loop
- Os command processor require infinite loops
- Body can also be empty
- Loop control var can be declared in the loop ()
- Scope of such var is only within that loop then var cease to exists

While loop

While (condition) stmt;

Statement can be single or block

Condition is checked at top of loop

Statement may not be executed at all

Do - while

Do {   } while(condition)

{ } not necessary when only one stmt is present

Providing { } gives readability

Break

- To exit from loop bypassing remaining stmt in the body & condition
- Can be used in any loop of Java
- Break out of innermost loop and outer loop is unaffected
- More than one break can appear

- Java does not have goto as the prog become hard to understand & maintain
- Break can be used to simulate goto

  Break label
- Label is name of block
- Cannot use break to transfer out of block that does not have break

Continue

- To force early iteration of loop bypassing normal control structure
- It is complement of break
- In case of while & do-while: Control goes directly to condition
- In case of for loop : update is done & then condition is tested
- Continue can have label to describe which enclosing loop to continue