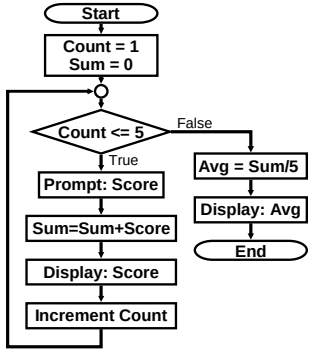
Flow of Control

- Flow of control
  - Definition: The sequence that computer executes program statements
- Sequential Control Structure
- Selection (Branching) Control Structure
  - Relational and Logical Operators
- Repetition (Loop) Control Structure
  - while loops
  - for loops

 1

Repetition (Loop) Structure

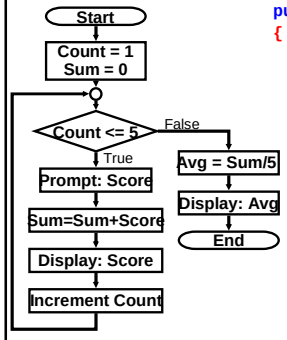
- Control structure used to repeat a sequence of instructions in a loop.



 2

Repetition (Loop) Structure

- Flowchart is translated to Java code
- The simplest loop structure is the while()




```
public static void main(String[] args)
{
  int nScore, nCount = 1, nSum = 0;
  String sEntry;
  while(nCount <= 5)
  {
    sEntry=JOptionPane.showInputDialog(
      null,"Enter Score "+nCount+":");
    nScore = Integer.parseInt(sEntry);
    nSum += nScore;
    System.out.println("Score "
      +nCount+" = " +nScore);
    nCount++;
  }
  System.out.println("\nAverage = "+nSum/5);
  System.exit(0);
}
```

 3

while statement loop control

- Contents of loop executed repeatedly while(*assertion)* is true
- Loop terminated when while(assertion) is false.
- Counter-Controlled Repetition Structure
  - Initialize a counter to count loops
  - Increment or decrement counter
  - while(assertion) checks for total loops reached
- Sentinel-Controlled Repetition Structure
  - while(assertion) checks for a sentinel termination value

 4

Counter-Controlled Repetition Structure

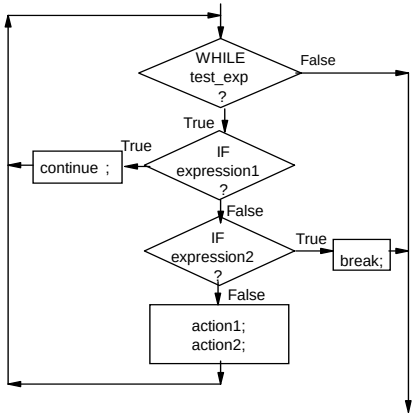
```
import javax.swing.*;
public class Example2
{
  public static void main(String[] args)
  {
    int nScore, nCount = 1, nSum = 0;
    String sEntry;
    while(nCount <= 5)
    {
      sEntry = JOptionPane.showInputDialog(
        null, "Enter Score "+nCount+":");
      nScore = Integer.parseInt(sEntry);
      nSum += nScore;
      System.out.println("Score "+nCount
        +" = " +nScore);

      nCount++;
    }
    System.out.println("\nAverage = "+nSum/5);
    System.exit(0);
  }
}
```

```
Score 1 = 78
Score 2 = 82
Score 3 = 87
Score 4 = 93
Score 5 = 86

Average = 85
```

break; continue; commands




```
while(test_exp)
{
    if(expression1)
      continue;
    if(expression2)
      break;
    action1;
    action2;
}
```


Sentinel-Controlled Repetition Structure

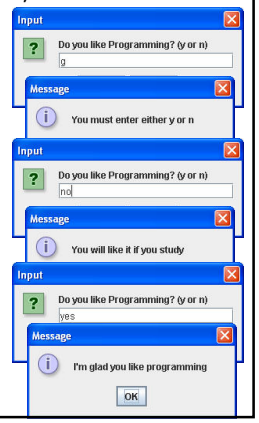
```
import javax.swing.*;
public class Example3
{
  public static void main(String[] args)
  {
    int nScore, nCount = 0, nSum = 0;
    String sEntry;
    while(true)
    {
      sEntry = JOptionPane.showInputDialog(
        null, "Enter Score "+(nCount+1)+":");
      nScore = Integer.parseInt(sEntry);
      if(nScore < 0)break;
      nSum += nScore;
      System.out.println("Score "+(nCount+1)
        +" = " +nScore);
      nCount++;
    }
    System.out.println("\nAverage = "+nSum/nCount);
    System.exit(0);
  }
}
```

```
Score 1 = 90
Score 2 = 80
Score 3 = 70
Score 4 = 60
Score 5 = 85
Score 6 = 65
Score 7 = 95
Score 8 = 55

Average = 75
```

Filtered Input Application

```
public static void main(String[] args)
{
  String sEntry;
  char cSelect;
  while(true)
  {
    sEntry = JOptionPane.showInputDialog(null,
      "Do you like Programming? (y or n)");
    cSelect = sEntry.charAt(0);
    if(cSelect == 'y')
    {
      JOptionPane.showMessageDialog(null,
        "I\'m glad you like programming");
      break;
    }
    else if(cSelect == 'n')
    {
      JOptionPane.showMessageDialog(null,
        "You will like it if you study");
      break;
    }
    else
      JOptionPane.showMessageDialog(null,
        "You must enter either y or n");
  }
  System.exit(0);
}
```



Exception Handling

- Exception Handling Processing
  - Error occurs while method is running
  - Method creates *Information Object* about error
  - Information Object passed to Java Virtual Machine
  - JVM attempts to locate code to handle exception
  - This process is called *Throwing an Exception*
- Checked exception
  - `try`
    - Identifies start of exception handling block of code
    - Must be followed by one or more catch blocks
  - `catch`
    - Exception handler code
  - `finally`
    - Default set of instructions that is always executed whether or not any exception occurred

Exception Handling

```
import javax.swing.*;
public class Exception2
{
  public static void main(String args[])
  {
    double dNum1,  dNum2;
    String sEntry, sDisplay;
    try
    {
      sEntry = JOptionPane.showInputDialog("Enter 1st Number:");
      dNum1 = Double.parseDouble(sEntry);
      sEntry = JOptionPane.showInputDialog("Enter 2nd Number:");
      dNum2 = Double.parseDouble(sEntry);
      sDisplay = "Operators Output\nby Bob Laurie\n"
        + dNum1+" + "+dNum2+" = "+(dNum1+dNum2)+'\n'
        + dNum1+" - "+dNum2+" = "+(dNum1-dNum2)+'\n'
        + dNum1+" / "+dNum2+" = "+(dNum1/dNum2)+'\n'
        + dNum1+" x "+dNum2+" = "+(dNum1*dNum2);
      JOptionPane.showMessageDialog(null, sDisplay);
    }
    catch(NumberFormatException n)
    {
      JOptionPane.showMessageDialog(null, "ERROR: Enter Number!");
    }
    catch(NullPointerException n)
    {
      JOptionPane.showMessageDialog(null, "ERROR: Cancel Pushed!");
    }
    finally
    {
      System.exit(0);
    }
  }
}
```

Mathematical Methods

- Java provides standard preprogrammed methods within class named `Math`
  - Methods are `static` and `public`
- Each `Math` class method is called by:
  - `returnValue Math.method(parameters);`
  - `dOutput Math.sqrt(dInput);`
  - `dOutput Math.pow(dBase, dExponent);`
  - `dOutput Math.abs(dInput);`
  - `dOutput Math.random(); // between 0. 1.`

Casts

- Java provides for explicit user-specified type conversions
- Use cast operator
  - Unary operator
  - Syntax:
    - `(dataType) expression`
  - Example:
    - `(int) (a * b)`
    - `(double) (a * b)`
    - `(short) (a * b)`
    - `(float) (a * b)`

