AP Computer Science A

Java Programming Essentials [Ver.4.0]

Unit 2: Selection and Iterations

CHAPTER 6: BOOLEAN CONDITIONS AND SELECTION

DR. ERIC CHOU
IEEE SENIOR MEMBER

AP Computer Science Curriculum

- Boolean Value, Boolean Value and Boolean Expression (T 2.2)
- Algorithms with Selection and Repetition (T 2.1)
- if Statements (T 2.3)
- Nested if Statements (T 2.4)

Objectives

- If Statement
- Random Number Generation
- Conditional Statement
- Switch Statement
- Simulation mode (Random Input)
- Simulation of Data Vectors in Continuous Domain

If Statement

Lecture 1

The `boolean` Type and Operators

- Often in a program you need to compare two values, such as whether i is greater than j. Java provides six comparison operators (also known as relational operators) that can be used to compare two values. The result of the comparison is a Boolean value: true or false.

```
boolean b = (1 > 2);
```

Boolean Data Type

The Boolean data type declares a variable with the value either true or false.

Relational Operators				
Java Operator	Math Symbol	Name	Example	Result
<	$<$	Less than	radius < 0	false
<=	$\leq$	Less than or Equal to	radius <= 0	false
>	$>$	Greater than	radius > 0	true
>=	$\geq$	Greater than or equal to	radius >= 0	true
==	$=$	Equal to	radius == 0	false
!=	$\neq$	Not Equal to	radius != 0	true

Boolean literals: **true** and **false.** These are the only values that will be returned by the Boolean expressions.

if statement

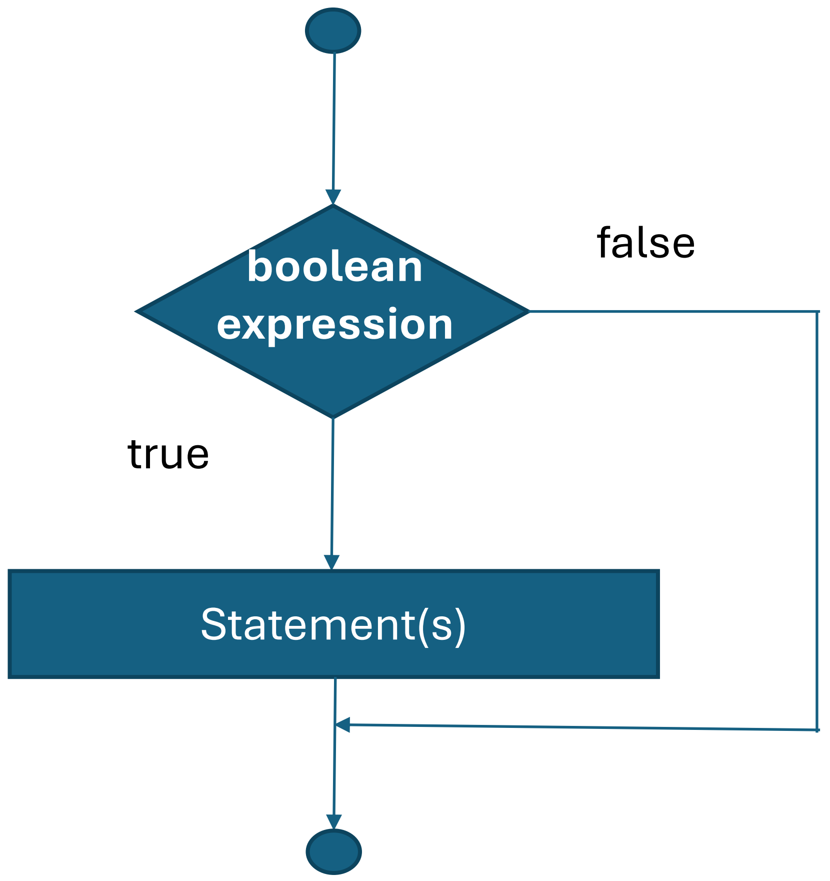
An if statement is a construct that enables a program to specify alternative paths of execution.

One way if statement:

```
if (Boolean-expression) {
  statement(s);
}
```

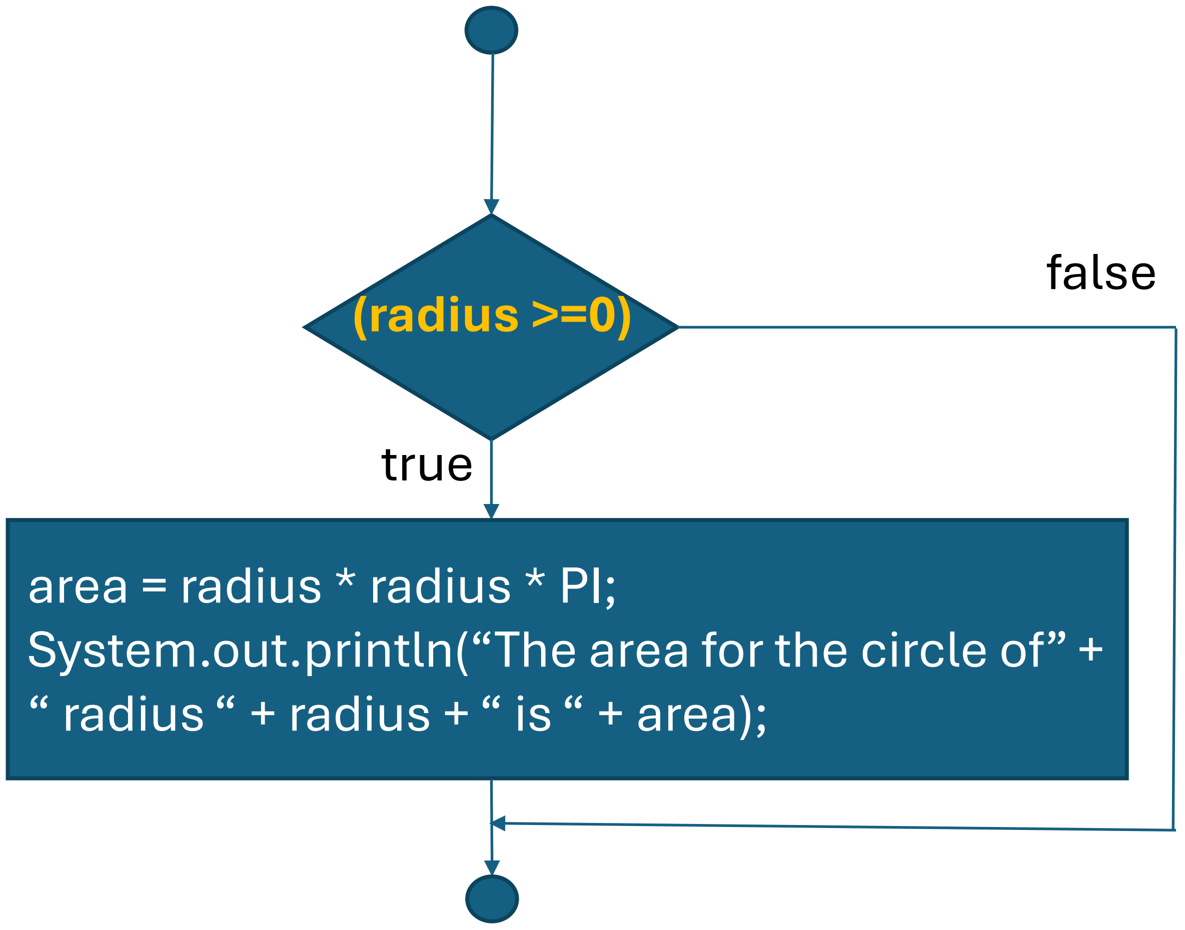
If there is only single statement: **if (i>0) i++;** is equivalent to

```
if (i>0) {
  i++;
}
```



if statement

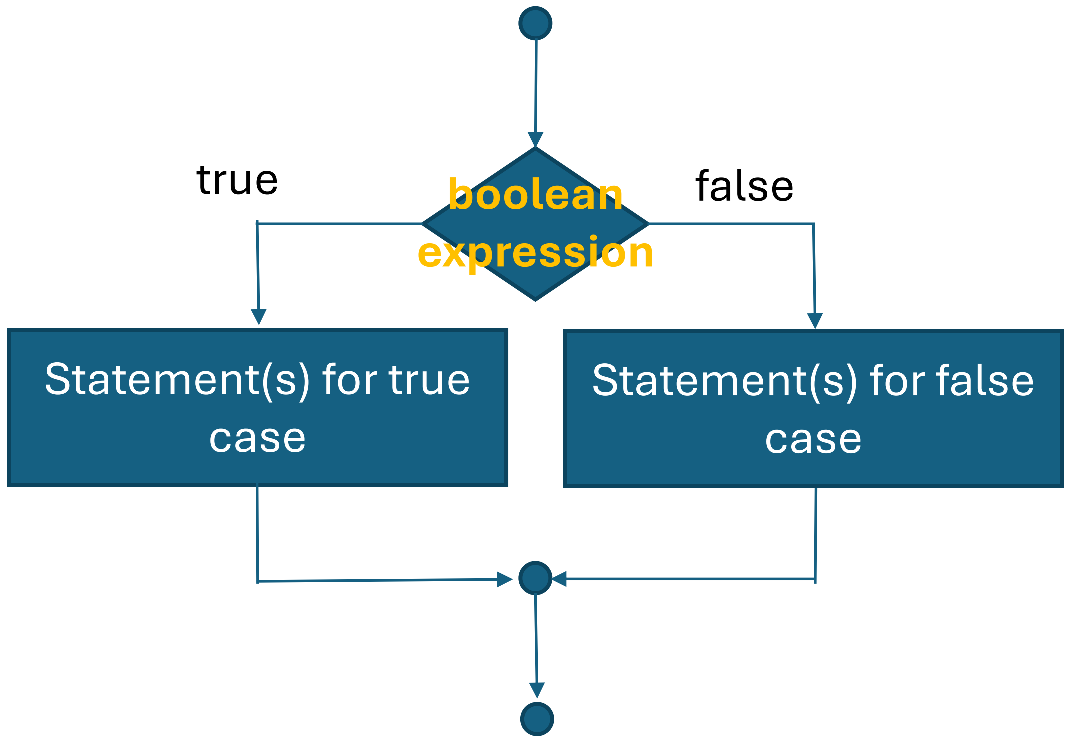
An if statement is a construct that enables a program to specify alternative paths of execution.


(radius >=0)
false
true
area = radius * radius * PI;
System.out.println("The area for the circle of" +
" radius " + radius + " is " + area);

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("The area"
    + " for the circle of radius "
    + radius + " is " + area);
}
```

Two-way if-else statements

An if-else statement decides the execution path based on whether the condition is true or false.



```
if (boolean-expression) {
    statement(s)-for-the-true-case;
}
else {
    statement(s)-for-the-false-case;
}
```

`if-else` Example

```
if (radius >= 0) {
  area = radius * radius * 3.14159;

     System.out.println("The area for the “
    + “circle of radius " + radius +
    " is " + area);
}
else {
  System.out.println("Negative input");
}
```

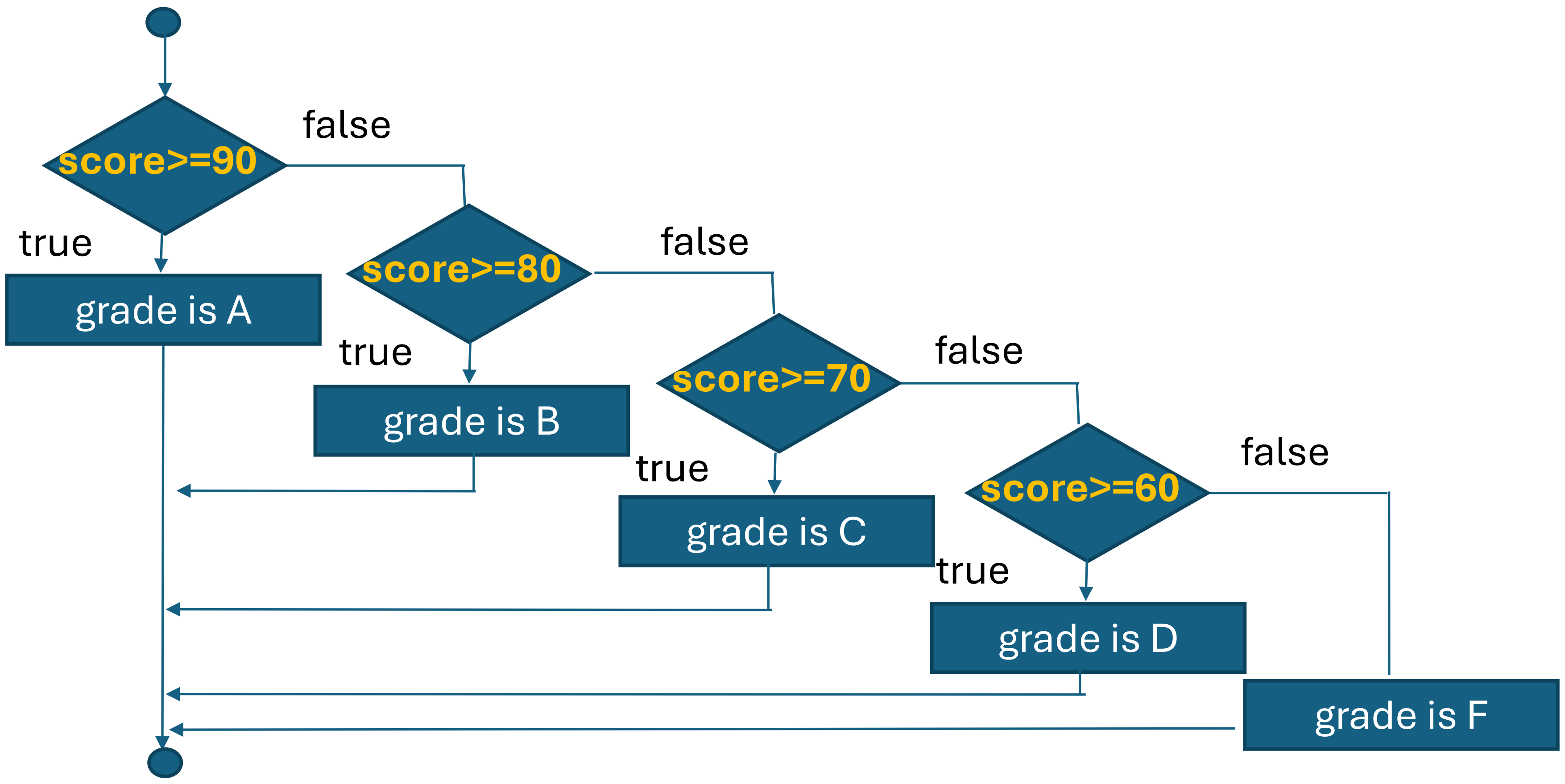
Nested if and Multi-Way if-else Statements

An if statement can be inside another ***if*** statement to form a nested ***if*** statement.

- The statement in an **if** or **if-else** statement can be any legal Java statement, including another **if** or **if-else** statement.
- The inner **if** statement is said to be nested inside the outer **if** statement.

Nested if and Multi-Way if-else Statements

An if statement can be inside another ***if*** statement to form a nested ***if*** statement.



Nested if and Multi-Way if-else Statements

An if statement can be inside another ***if*** statement to form a nested ***if*** statement

```
if (score>=90.0)
  System.out.print(“A”);
else if (score>=80.0)
  System.out.print(“B”);
else if (score>=70.0)
  System.out.print(“C”);
else if (score>=60.0)
  System.out.print(“D”);
else
  System.out.print(“F”);
```

```
if (score>=90.0)
  System.out.print(“A”);
else if (score>=80.0)
          System.out.print(“B”);
      else if (score>=70.0)
               System.out.print(“C”);
            else if (score>=60.0)
                     System.out.print(“D”);
                 else
                     System.out.print(“F”);
```

Note

The else clause matches the most recent if clause in the same block.

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    System.out.println("A");
else
    System.out.println("B");
```

(a)

Equivalent

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
  if (i > k)
    System.out.println("A");
  else
    System.out.println("B");
```

(b)

Note

- Nothing is printed from the preceding statement. To force the else clause to match the first if clause, you must add a pair of braces:

```
int i = 1;
int j = 2;
int k = 3;
if (i > j) {
  if (i > k)
    System.out.println("A");
}
else
  System.out.println("B");
```

This statement prints B.

TIP

```
if (number % 2 == 0)
  even = true;
else
  even = false;
```

(a)

Equivalent

```
boolean even
  = number % 2 == 0;
```

(b)

CAUTION

```
if (even == true)
  System.out.println(
    "It is even.");
```

(a)

Equivalent

```
if (even)
  System.out.println(
    "It is even.");
```

(b)

Common Errors and Pitfalls

- Common Error 1: Forgetting Necessary Braces.
  - **if (logical expr.) a++; b++; c++;**
- Common Error 2: Wrong Semicolon at the if Line.
  - if (logical expr.) **{}**; {.....}
- Common Error 3: Redundant Testing of Boolean Values.
  - if (**even == true**) ... when even is a **Boolean** variable.
- Common Error 4: Dangling else Ambiguity.
  - **if (A) if (B) statements; else statement; (Use parenthesis or combine the two)**

Common Errors and Pitfalls

Common Error 5: Equality Test of Two Floating Point Values.

- **Floating point is not accurate can not be tested for equality. It will cause trouble.**

Common Pitfall 1: Simplifying Boolean Variable Assignment

- Use Boolean assignment to replace if statement: **boolean even = number % 2 == 0;   (mod(%), equality check have higher priority than assignment)**

Common Pitfall 2: Avoiding Duplicate Code in Different Case.

- A statement of same result is put in both if and else cases.

Demonstration Program

SimpleifDemo.java

Lab

Lab: Body Mass Index

Background Information

- Your BMI is a measurement of your body weight based on your height and weight. Although your BMI does not actually "measure" your percentage of body fat, it is a useful tool to estimate a healthy body weight based on your height. Due to its ease of measurement and calculation, it is the most widely used diagnostic indicator to identify a person's optimal weight depending on his height.
- Your **BMI** "number" will inform you if you are underweight, of normal weight, overweight, or obese. However, due to the wide variety of body types, the distribution of muscle and bone mass, etc., it is not appropriate to use this as the only or final indication for diagnosis.

Body Mass Index Formula

- The formulas to calculate BMI based on two of the most commonly used unit systems:
- **BMI = weight(kg)/height^2(m^2)   (Metric Units)**
  BMI = 703·weight(lb)/height^2(in^2)   (U.S. Units)
-

BMI Table for Adults

Category	**BMI range - kg/m^2**
Severe Thinness	< 16
Moderate Thinness	16 - 17
Mild Thinness	17 - 18.5
Normal	18.5 - 25
Overweight	25 - 30
Obese Class I	30 - 35
Obese Class II	35 - 40
Obese Class III	> 40

For More Information, Please Check

- **http://www.calculator.net/bmi-calculator.html**

Lab: BMI.java

- Write a program to ask for USER to enter his/her (adult) name, weight (kg), and height (m), to calculate for his BMI value. Then, print out a BMI report for this person including the following information:

Name:

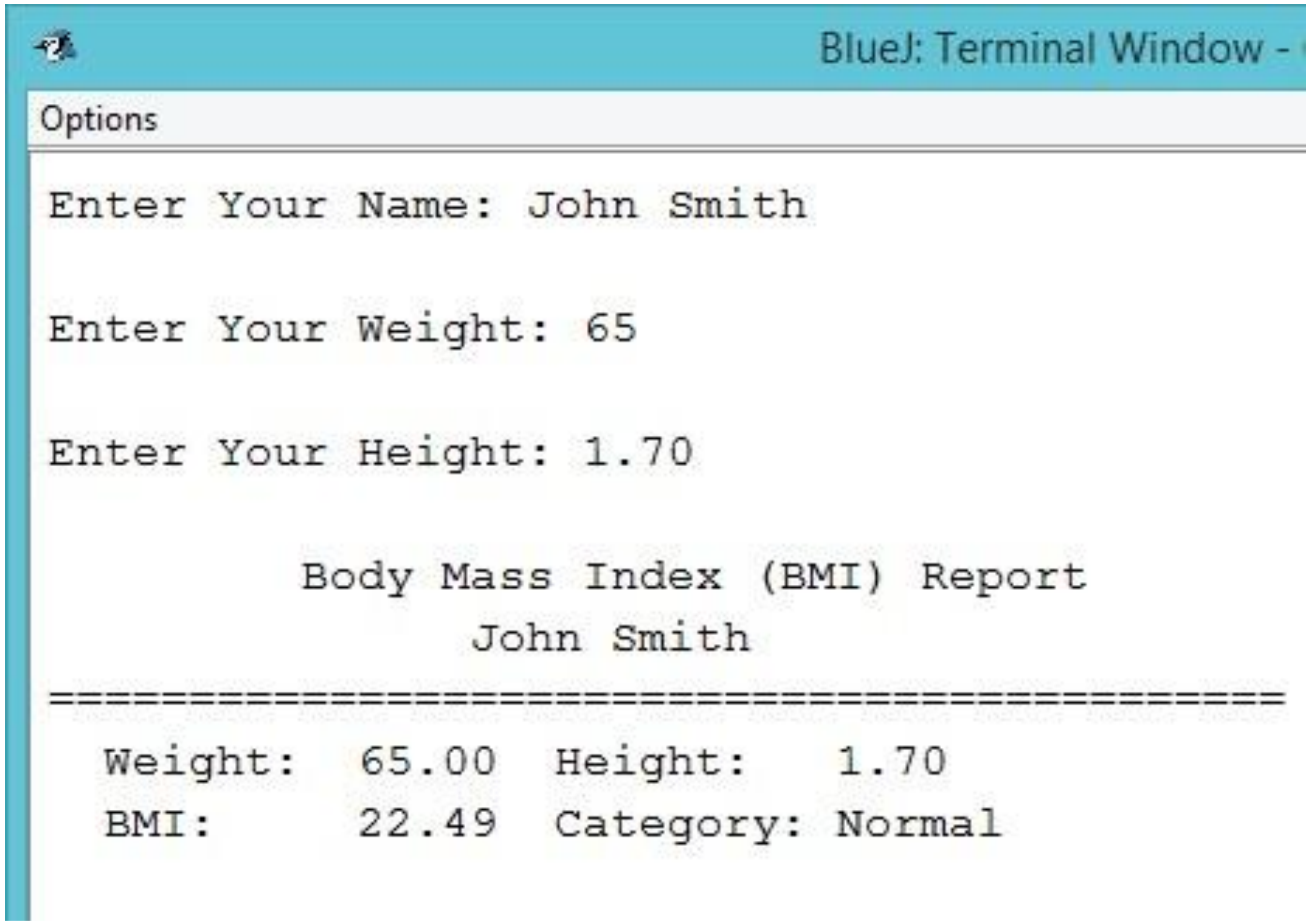
Weight: (2 decimal place)

Height: (2 decimal place)

BMI: (2 decimal place)

BMI Category:

Expected Result:



```
Enter Your Name: John Smith

Enter Your Weight: 65

Enter Your Height: 1.70

          Body Mass Index (BMI) Report
                 John Smith
============================================
  Weight:   65.00  Height:    1.70
  BMI:      22.49  Category: Normal
```

Demonstration Program

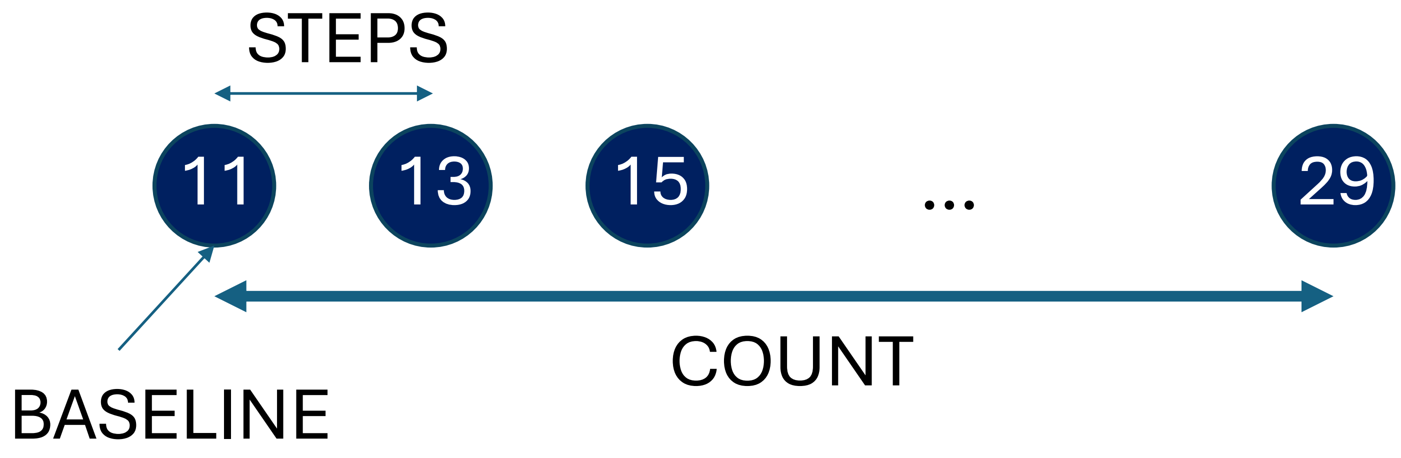
StudentGPAWithLetterGrade.java

Random Number Generation

Lecture 2

Review of Random Sample Generation

- int r = ((int) (Math.random()*COUNT) * STEPS + BASELINE;
- **int r = ((int) (Math.random() * 10) * 2 + 11);**



Generation of a Real Number Range

- double r = (Math.random()*range) + BASELINE;
- [Baseline, Baseline+range)

- [Baseline, Baseline+range]
- double y=-999;
-         while (y<-Baseline || y>Baseline+range){
-             y = Math.random()*(range+0.1)-range/2;
-         }

Random Number Generation for Different Sample Regions **(weighted randomization)**

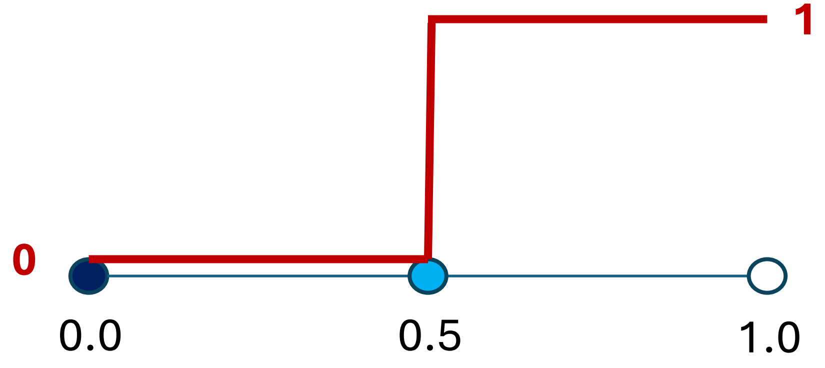
- **Generation Random Alphanumerical Symbols:**
- int region = (int) (Math.random()* 3);
- char dLetter = (char) (Math.random()*10);
- char aLetter = (char) (Math.random()*26);
- if (region == 0) aLetter = (char) (dLetter + ‘0’);
- else if (region == 1) aLetter += ‘A’;
- else aLetter += ‘a’;
- // at the end, aLetter will be a random alphanumerical symbol.

Random Number Generation for Different Sample Regions **(unweighted randomization)**

- **Generation Random Alphanumerical Symbols:**
- char aLetter = (char) (Math.random()* 62);
- if (aLetter <= 9) aLetter += ‘0’; // 0 to 9
- else if (aLetter <= 35) aLetter = aLetter - (char) 10 + ‘A’; // 10 - 35
- else aLetter = aLetter – (char) 36 + ‘a’; // 36 – 61
- // at the end, aLetter will be a random alphanumerical symbol.

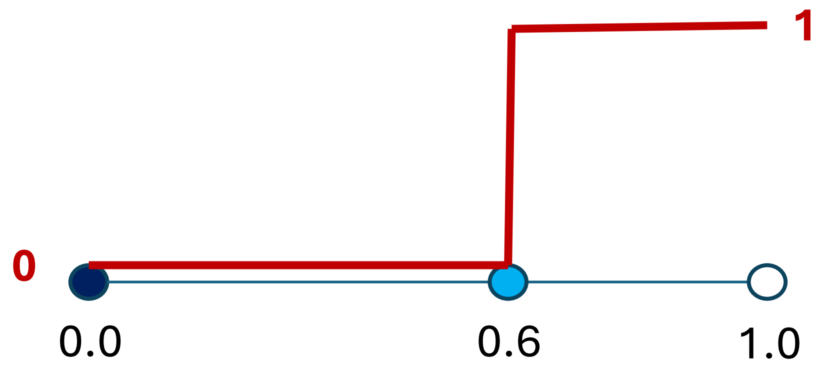
Un-biased Randomized Coin (50-50)

- Unbiased Random Number Generator:
- double randToss = Math.random();
- int die = 1;
- if (randToss < 0.5) die = 0; // preset-else
- // Think about it, you do not need the else-part.
- // another way to write it.
- int ide = (randToss<0.5) ? 0 : 1;
- // conditional expression.
- //(coming lecture in this chapter)



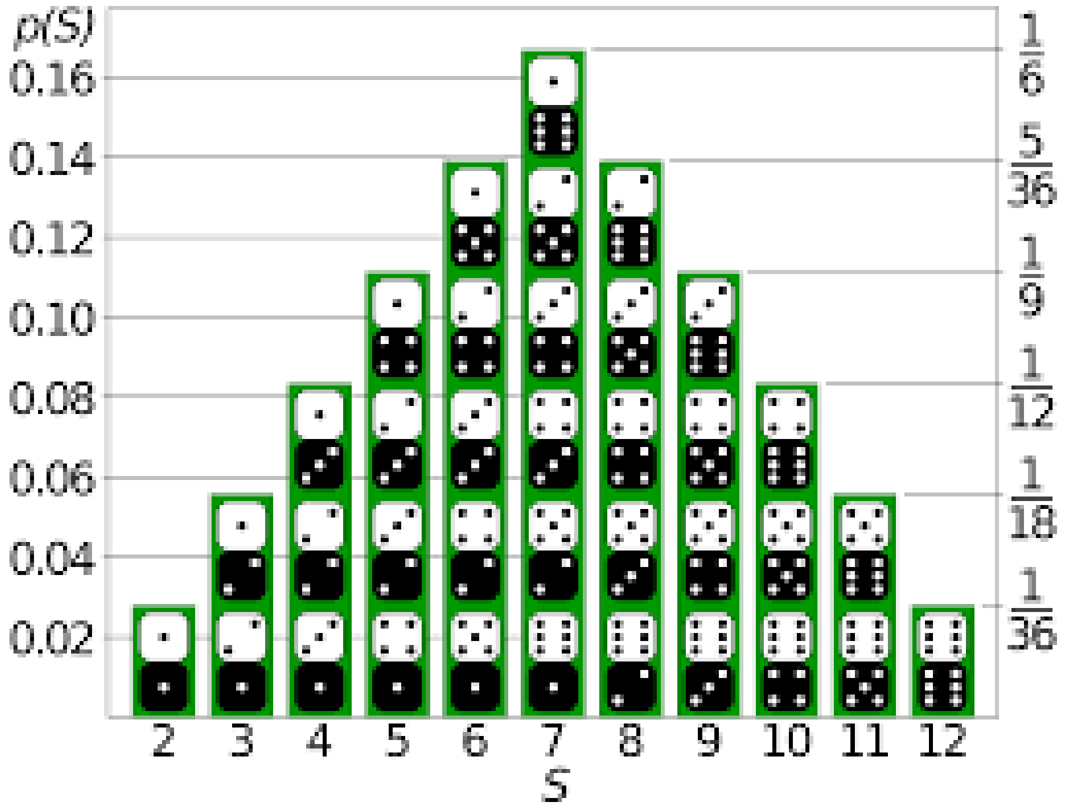
Biased Randomized Coin (60% - 0 (Tail))

- Unbiased Random Number Generator:
- double randToss = Math.random();
- int die = 1;
- if (randToss < 0.6) die = 0; // preset-else
- // Think about it, you do not need the else-part.
- // another way to write it.
- int ide = (randToss<0.6) ? 0 : 1;
- // conditional expression.
- //(coming lecture in this chapter)



Sum of Two Dice Randomized Test

- int die1 = (int) (Math.random()*6) + 1;
- int die2 = (int) (Math.random()*6) + 1;
- int sum = die1 + die2;



Why Random Number Generator is so Important in Computer Science?

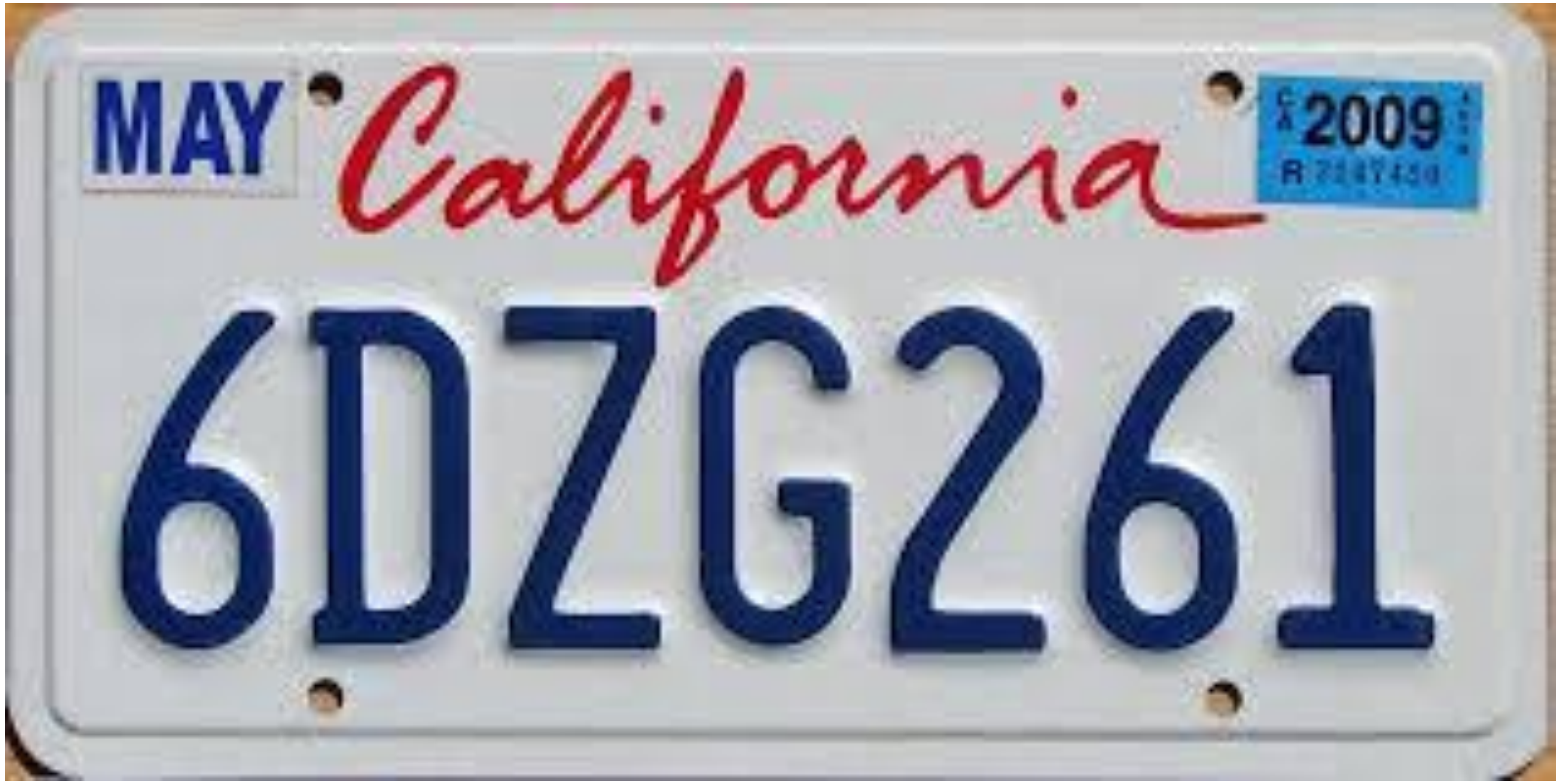
Applications:

- Computer Game Design
- Monte Carlo Simulation (Computer Simulation)
- Use Simulation to Solve Hard Problems by Analytical Modelling
- Reality Check for a Project before Implementation

Lab

Lab: Californian State DMV

Californian State License Plate for Cars



Rules

- 3 sections:
- Leading digit: following state assignment, use 7 right now.
- Letter code (3): No repeating letters;
- Digit code (3): No repeating digits.

Lab: LicensePlate.java

- Write a program to generate a license plate number follows the Californian State DMV’s rules. (No repeating uppercase letter and digits);

- Hint: Use the following statement to create a digit or a letter

char dLetter = (char) ((int) (Math.random()*10) + ‘0’);

- char aLetter = (char) ((int) (Math.random()*26) + ‘A’);

Lab: LicensePlate.java

- Hint:

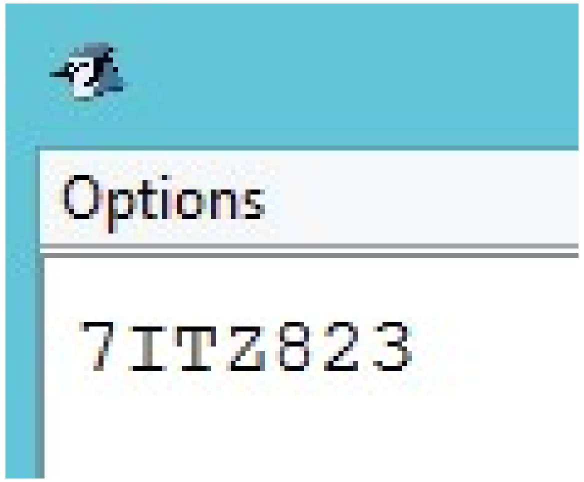
(1) concatenate all digits and letters together to create the License plate numbers.

(2) keep a copy of the digits and letters that has been used to avoid repeating them.

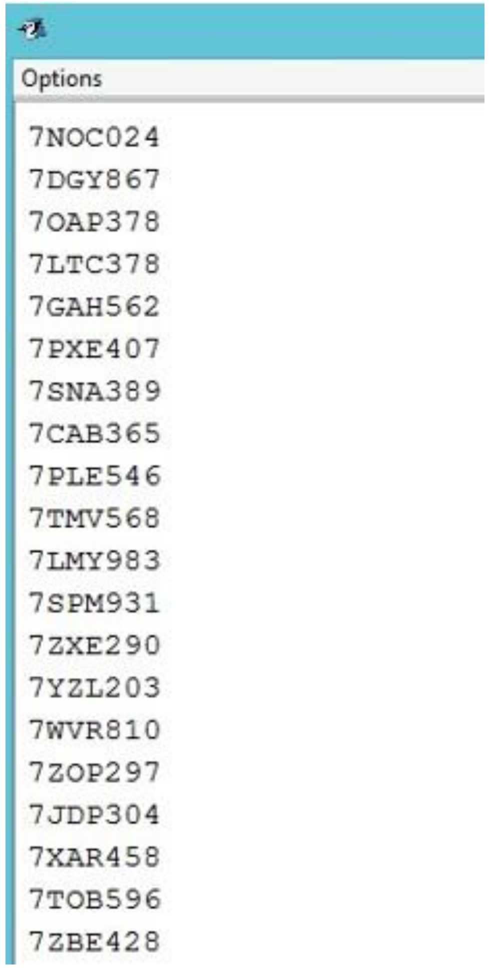
```
// create letter2 to be non-repeating compared with letter1
char letter2 = (char)((int)(Math.random()*26)+'A');
if (letter1 == letter2) letter2 = (char)((letter2-'A' + 1) % 26 + 'A');
```

(3) No loops in this Lab work.

Expected Result:


Options

```
7ITZ823
```


Options

```
7NOC024
7DGY867
7OAP378
7LTC378
7GAH562
7PXE407
7SNA389
7CAB365
7PLE546
7TMV568
7LMY983
7SPM931
7ZXE290
7YZL203
7WVR810
7ZOP297
7JDP304
7XAR458
7TOB596
7ZBE428
```

Conditional Statements

Lecture 3

Conditional Expressions

A conditional expression evaluates an expression based on a condition.

Conditional Statement Syntax:

boolean-expression ? expression_true_case : expression_false_case;

true : **false**

If-then-else statement:

```
if (  x > 0 )
   y = 1;
else
  y = -1;
```

Conditional statement:

```
y = (x > 0) ?     1    :    -1      ;
                true  :  false
```

Ternary operator
Binary operator
Unary operator

Conditional Operator

```
if (num % 2 == 0)
  System.out.println(num + “is even”);
else
  System.out.println(num + “is odd”);


System.out.println(
  (num % 2 == 0)? num + “is even” :
  num + “is odd”);
```

Switch Statements

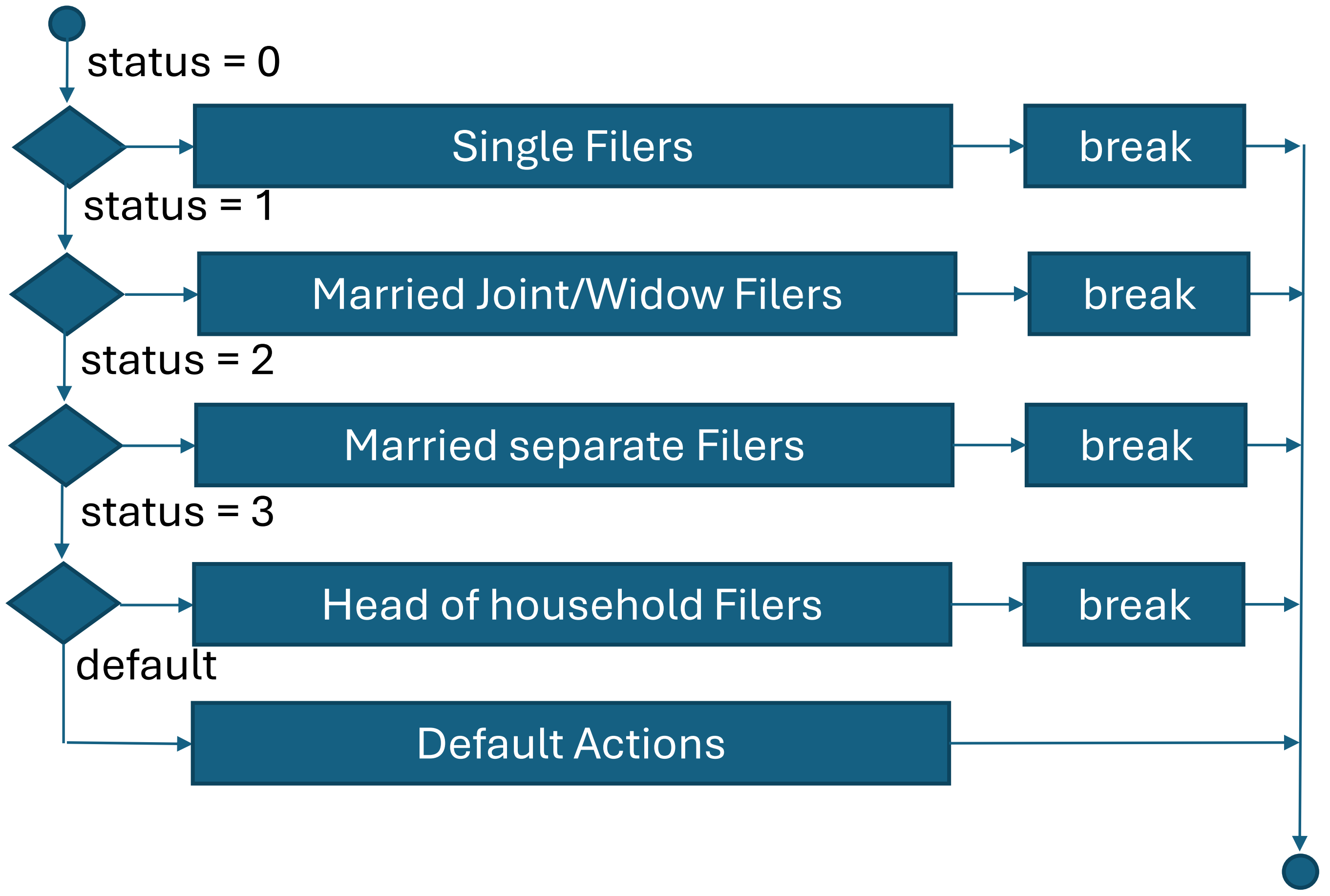
Lecture 4

switch Statements

A switch statement executes statements based on the value of a variable or an expression

```
switch (status) {
  case 0: compute tax for single filers;
             break;
  case 1: compute tax for married filing jointly or qualifying widow(s);
             break;
  case 2: compute tax for married filing separately;
             break;
  case 3: compute tax for head of household;
             break;
  default: System.out.println("Error: invalid status");
             System.exit(1);
}
```

Flowchart for Tax status in switch statement.



switch Statement Rules

The switch-expression must yield a value of **char, byte, short, int** or **string type** and must always be enclosed in parentheses.

The value1, ..., and valueN must have the same data type as the value of the switch-expression. The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression. Note that value1, ..., and valueN are constant expressions, meaning that they cannot contain variables in the expression, such as 1 + x.

```
switch (switch-expression) {
 case value1:  statement(s)1;
       break;
 case value2: statement(s)2;
       break;
 …
 case valueN: statement(s)N;
       break;
 default: statement(s)-for-
default;
}
```

switch Statement Rules

The keyword break is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement. If the break statement is not present, the next case statement will be executed.

```
switch (switch-expression) {
  case value1:  statement(s)1;
        break;
  case value2: statement(s)2;
        break;
  …
  case valueN: statement(s)N;
        break;
  default: statement(s)-for-default;
}
```

The default case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.

The case statements are executed in sequential order, but the order of the cases (including the default case) does not matter. However, it is good programming style to follow the logical sequence of the cases and place the default case at the end.

Chinese Zodiac in switch format (Continues from Lab work.)

```
switch (zodiacYear) {
  case  0:  zodiac = "Rat";       break;
  case  1:  zodiac = "Ox";        break;
  case  2:  zodiac = "Tiger";     break;
  case  3:  zodiac = "Rabbit";    break;
  case  4:  zodiac = "Dragon";    break;
  case  5:  zodiac = "Snake";     break;
  case  6:  zodiac = "Horse";     break;
  case  7:  zodiac = "Sheep";     break;
  case  8:  zodiac = "Monkey";    break;
  case  9:  zodiac = "Rooster";   break;
  case 10: zodiac = "Dog";        break;
  case 11: zodiac = "Pig";        break;
}
```

What will happen without breaks?

```
switch (day) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5: System.out.println(“Weekday”); break;
  case 0:
  case 6: System.out.println(“Weekend”);
}
```

Day 1-5, you will get Weekday
Day 0 and Day 6, you will get Weekend

Random Class for Simulated Inputs

Lecture 5

Introduction

The **java.util.Random** class instance is used to generate a stream of pseudorandom numbers.Following are the important points about Random:

- The class uses a **48-bit seed**, which is modified using a linear congruential formula.
- The algorithms implemented by class Random use a protected utility method that on each invocation can supply up to **32** pseudorandomly generated bits.

USE of Random Class similar to Scanner;

- **Import Package :**
- import java.util.Scanner;
- **Declare input Stream Handler:**
- Scanner input = new Scanner(System.in);
- **Get formatted data:**
- int num = input.nextInt();

Import Package :

import java.util.Random;

Declare random input Stream Handle

Random input = new Random(seed);

Get formatted data:

int num = input.nextInt(100);

Constructor for Random Class

S.N.	Constructor & Description
1	**Random()** This creates a new random number generator.
2	**Random(long seed)** This creates a new random number generator using a single long seed.

S.N.	Method & Description
1	protected int next(int bits) This method generates the next pseudorandom number.
2	boolean nextBoolean() This method returns the next pseudorandom, uniformly distributed boolean value from this random number generator's sequence.
4	double nextDouble() **// similar to Math.random()** This method returns the next pseudorandom, uniformly distributed **double value between 0.0 and 1.0** from this random number generator's sequence.
5	float nextFloat() **// similar to Math.random()** This method returns the next pseudorandom, uniformly distributed **float value between 0.0 and 1.0** from this random number generator's sequence.
7	int nextInt() This method returns the next pseudorandom, uniformly distributed int value from this random number generator's sequence.
8	int nextInt(int n) **// similar to (int) (Math.random()*n)** This method returns a pseudorandom, uniformly distributed int value **between 0 (inclusive) and the specified value (exclusive)**, drawn from this random number generator's sequence.
9	long nextLong() This method returns the next pseudorandom, uniformly distributed long value from this random number generator's sequence.
10	void setSeed(long seed) This method sets the seed of this random number generator using a single long seed.

Example: nextInt(100); // 0-99

```
import java.util.Random;
/** Generate 10 random integers in the range 0..99. */
public final class RandomInteger {
  public static final void main(String... aArgs){
    System.out.println("Generating 10 random integers in range 0..99.");
    //note a single Random object is reused here
    Random randomGenerator = new Random();
    for (int idx = 1; idx <= 10; ++idx){
      int randomInt = randomGenerator.nextInt(100);
      log("Generated : " + randomInt);
    }
    System.out.println("Done.");
  }
}
```

Example run of this class:

```
Generating 10 random integers in range
0..99.
Generated : 44
Generated : 81
Generated : 69
Generated : 31
Generated : 10
Generated : 64
Generated : 74
Generated : 57
Generated : 56
Generated : 93
Done.
```

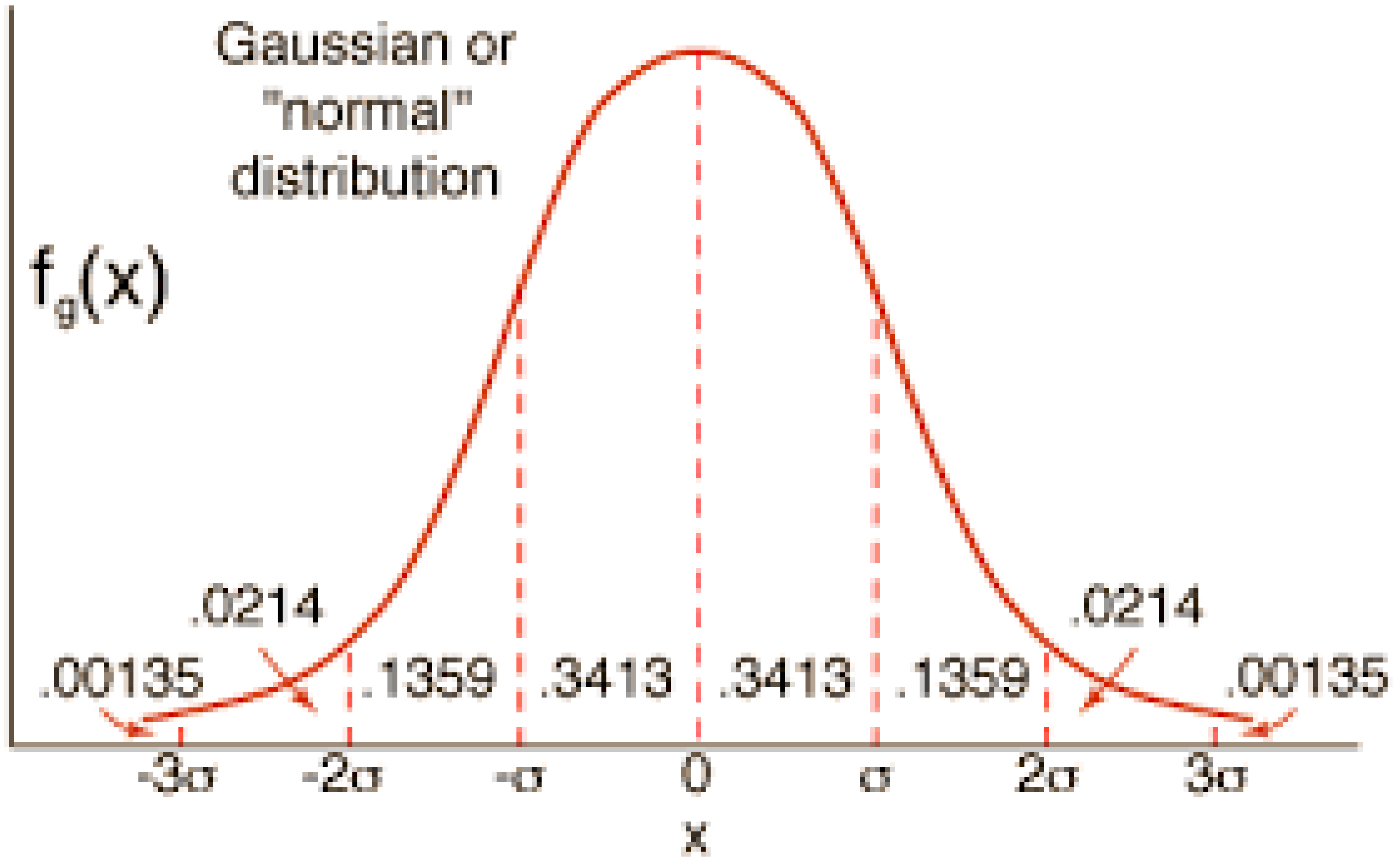
Random Floating Point Numbers in a Gaussian (Normal) Distribution.

```
import java.util.Random;
public final class RandomGaussian {
 public static void main(String... aArgs){
  RandomGaussian gaussian = new RandomGaussian();
  double MEAN = 100.0f;
  double VARIANCE = 5.0f;
  for (int idx = 1; idx <= 10; ++idx){
   log("Generated : " + gaussian.getGaussian(MEAN,
VARIANCE));
  }
```

Random Floating Point Numbers in a Gaussian (Normal) Distribution.

```
 private Random fRandom = new Random();
 private double getGaussian(double aMean, double
aVariance){
   return aMean + fRandom.nextGaussian() * aVariance;
 }
 private static void log(Object aMsg){
   System.out.println(String.valueOf(aMsg));
 }
}
```

Gaussian Distribution



Run Result:

An example run of this class:

```
Generated : 99.38221153454624
Generated : 100.95717075067498
Generated : 106.78740794978813
Generated : 105.57315286730545
Generated : 97.35077643206589
Generated : 92.56233774920052
Generated : 98.29311772993057
Generated : 102.04954815575822
Generated : 104.88458607780176
Generated : 97.11126014402141
```

Simulation Mode

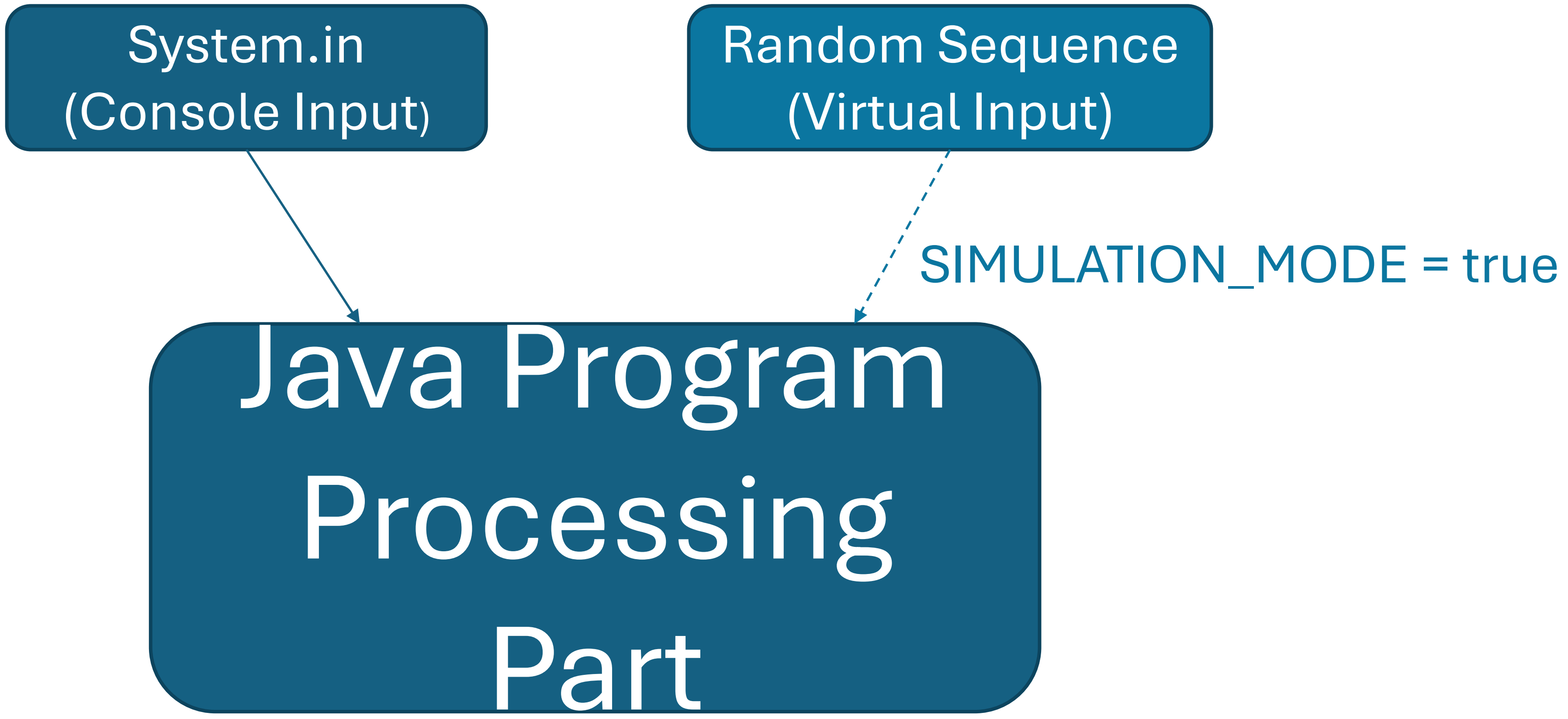
Lecture 6

Demo Program:
(StudentGPASimulationMode.java)

- Modified from StudentGPAWithLetterGrade.java
- **Updates:**
- (1) Use Random class for random input for simulation mode.
- (2) Random Seed for Random Number Generation.
- (3) Update the Logic design for full coverage for score input domain to handle the abnormal inputs.

Random Class: Another Input Stream

(Not really I/O but for simulation mode)



Demonstration Program

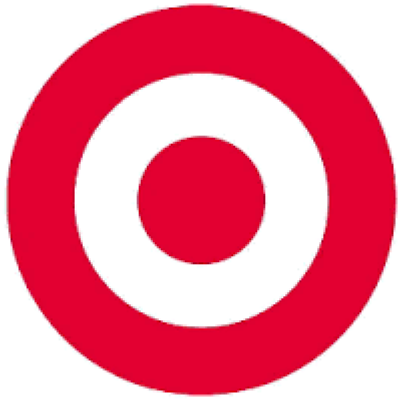
StudentGPASimulationMode.java

Lab Dart Game:

Lecture 7

Darts Game

- Darts throwing is a favorite game. It is fun and exciting.
- In this chapter project, we will be simulating such a game.
- Assume that a board of three circles, each has radius of 10cm, 20cm and 30cm respectively is used.



- Hitting the smallest circle scores 3 points.
- Hitting the middle circle scores 2 points
- Hitting the largest circle scores 1 points.
- Missing the board, no point.

Darts Game

- If a player throws three darts **evenly** over a 80cm x 80 cm area.
- Please answer the following questions:
- What is the probability of scoring 3 point on each throw?
- What is the probability of scoring 2 point on each throw?
- What is the probability of scoring 1 point on each throw?
- What is the probability of missing on each throw?
- Use your geometry knowledge to answer these questions.

Work Sheet

Total area = Area of largest circle =

Area of medium circle = Area of small circle =

Area for 0 point = Total area – Area of largest circle=

Area of 1 point = Area of largest circle - Area of medium circle =

Area of 2 points = Area of medium circle - Area of small circle =

Area of 3 points = Area of small circle =

Probability of 3 points = Area of 3 points / Total area =

Probability of 2 points = Area of 2 points / Total area =

Probability of 1 points = Area of 1 point / Total area =

Probability of 0 point = Area of 0 point / Total area =

Computer Simulation

- Write a program to simulate that a player throw 3 darts in a game. (MUST USE Random Class to generate random number, using nextDouble())
- First, use a random number generator to generate a dart's (x, y) coordination. Suppose that you throw a dart evenly over a square area of 80cm x 80cm.
- x from -40 to 40 (not including 40 and continuous value), and y from -40 to 40 as well. (units: cm)

Computer Simulation

- The distance from that dart’s location to the center of the circles is

$$d = distance\ (dart, center) = \sqrt{x^2 + y^2}$$

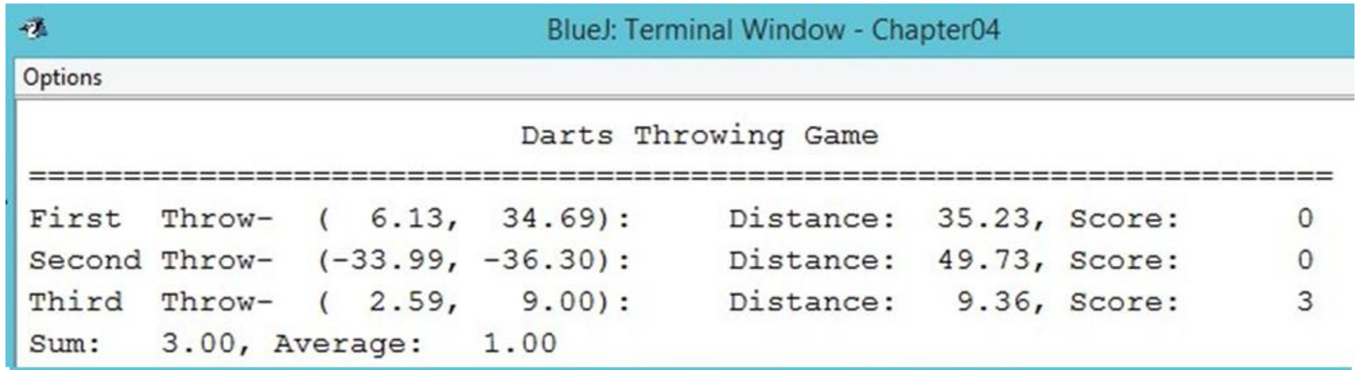
- $\boldsymbol{r_s}$, $\boldsymbol{r_m}$, and $\boldsymbol{r_b}$ are the radius of small circle, medium circle, and large circle respectively. ***s*** is the side of the square board of 80 cm a side with all circles centered at the center of the square board.

Scoring of the darts game

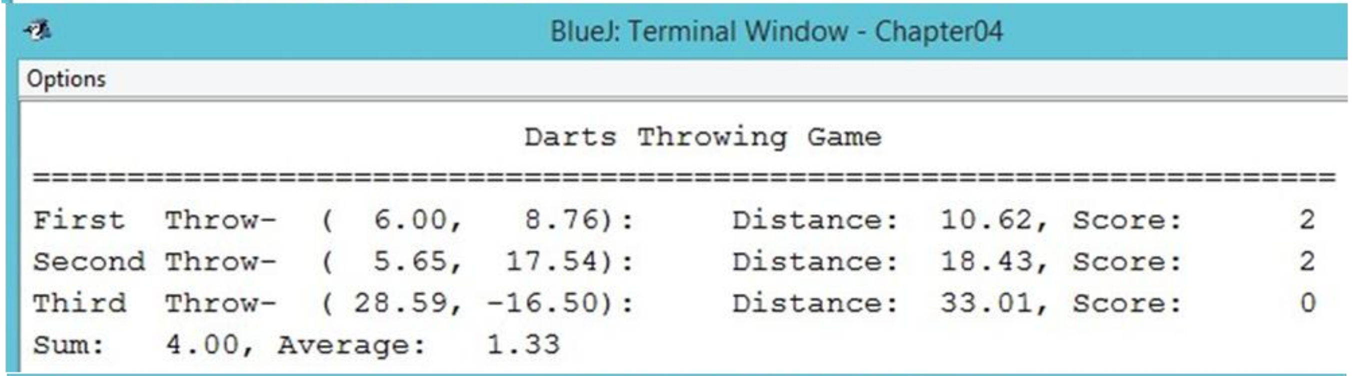
- If $\boldsymbol{d < r_s}$, then it is a 3-point throw.
- If $\boldsymbol{r_s < d < r_m}$, then it is a 2-point throw.
- If $\boldsymbol{r_m < d < r_b}$, then it is a 1-point throw.
- If $\boldsymbol{d > r_b}$, then it is a 0-point throw.

Simulate 3 throws for a player.

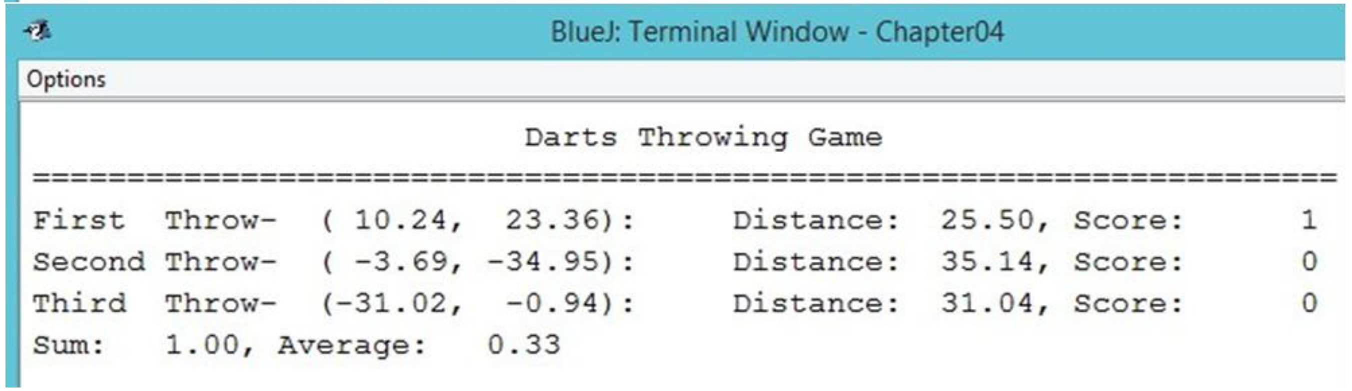
- Simulate 3 throws for a player,
- Print a Dart game report header.
- On each throw, print out its dart location (x, y), distance from center, and its score.
- At the end, print out the sum of score for 3 darts, and the average score for each darts.
- All numbers must be in proper formatted printf format (2 decimal places).
- We will revisit this program in the next chapter.



```
                        Darts Throwing Game
===============================================================
First  Throw-  (  6.13,   34.69):     Distance:  35.23, Score:      0
Second Throw-  (-33.99, -36.30):     Distance:  49.73, Score:      0
Third  Throw-  (  2.59,    9.00):     Distance:   9.36, Score:      3
Sum:   3.00, Average:   1.00
```



```
                        Darts Throwing Game
===============================================================
First  Throw-  (  6.00,    8.76):     Distance:  10.62, Score:      2
Second Throw-  (  5.65,   17.54):     Distance:  18.43, Score:      2
Third  Throw-  ( 28.59, -16.50):     Distance:  33.01, Score:      0
Sum:   4.00, Average:   1.33
```



```
                        Darts Throwing Game
===============================================================
First  Throw-  ( 10.24,   23.36):     Distance:  25.50, Score:      1
Second Throw-  ( -3.69, -34.95):     Distance:  35.14, Score:      0
Third  Throw-  (-31.02,   -0.94):     Distance:  31.04, Score:      0
Sum:   1.00, Average:   0.33
```

Expected Results: (sample answer program Darts.java)

Calculated Answers:

Prob. Of 0 point: 55.82%

Prob. Of 1 point: 24.55%

Prob. Of 2 points: 14.73%;

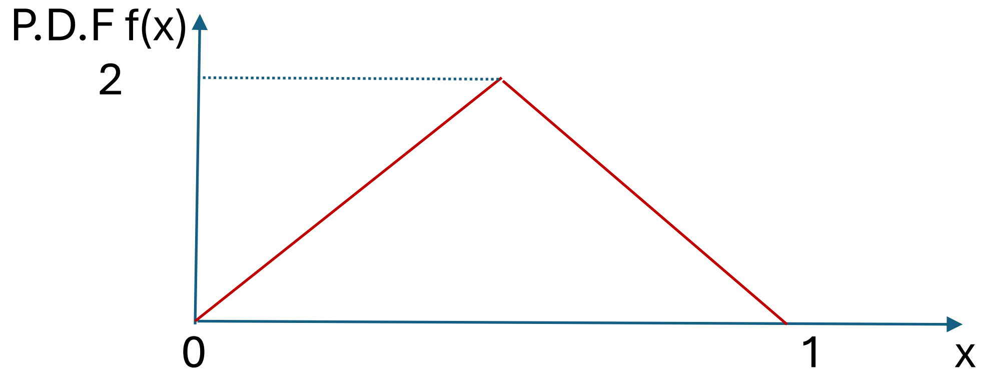
Prob. of 3 points: 4.92%

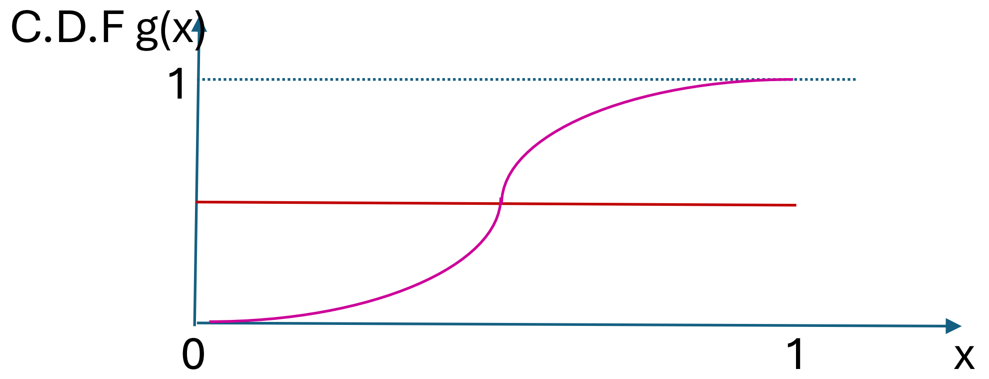
Advanced Topics on Random (Optional)

Lecture 8

Biased Random Sample (Advanced Topic)

Using Cumulative Distribution Function for Ramping Distribution Samples.



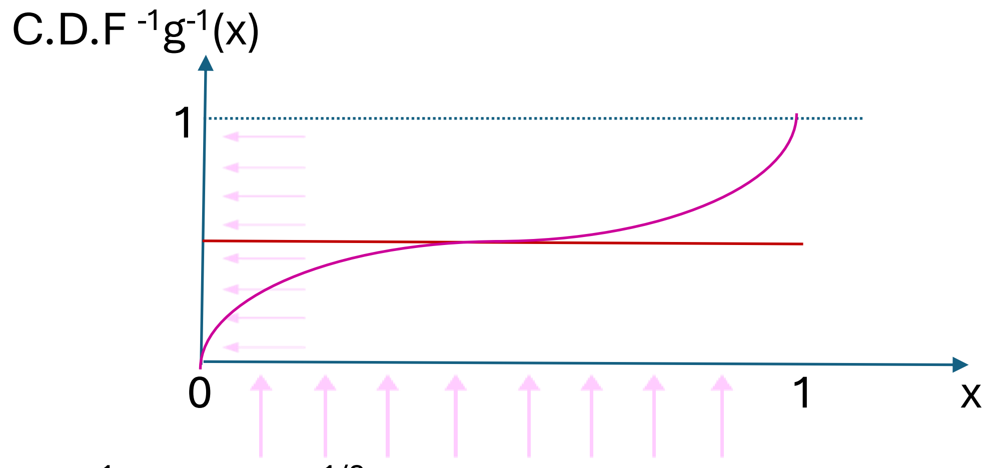


$f(x) = 4 * x$ if $0 <= x < 1/2$;
$f(x) = - 4 * (x- 1)$ if $1/2<= x < 1$

$g(x) = 2 * x^2$ if $0 <= x < 1/2$;
$g(x) = - 2(x -1)^2 + 1$ if $1/2<= x < 1$

Biased Random Sample (Advanced Topic)

Using Cumulative Distribution Function for Ramping Distribution Samples.



$g^{-1}(x) = (x/2)^{1/2}$ if $0 <= x < 1/2$;

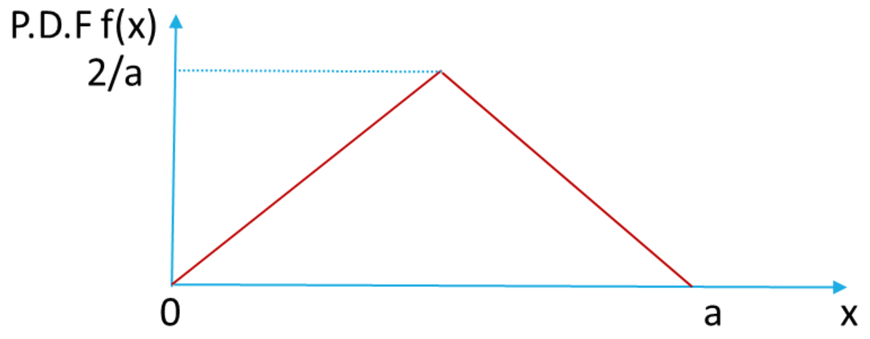
$g^{-1}(x) = 1.0 - ((1-x)/2)^{1/2}$ if $1/2 <= x < 1$

```
// Projection from random number
// generator to CDF-1 y = g-1(x),  y will have the
// probability distribution function of f(x)
(PDF)

double x = Math.random();
double y = 0.0;

if ( x>= 0 && x < 0.5) y = Math.pow(x/2.0, 0.5);
else y = 1.0-Math.pow((1-x)/2, 0.5);
```

Biased Random Sample (Advanced Topic)

Using Cumulative Distribution Function for Ramping Distribution Samples.

- To generate random number of arbitrary range:
- **double z = y * SPAN + BASELINE;**

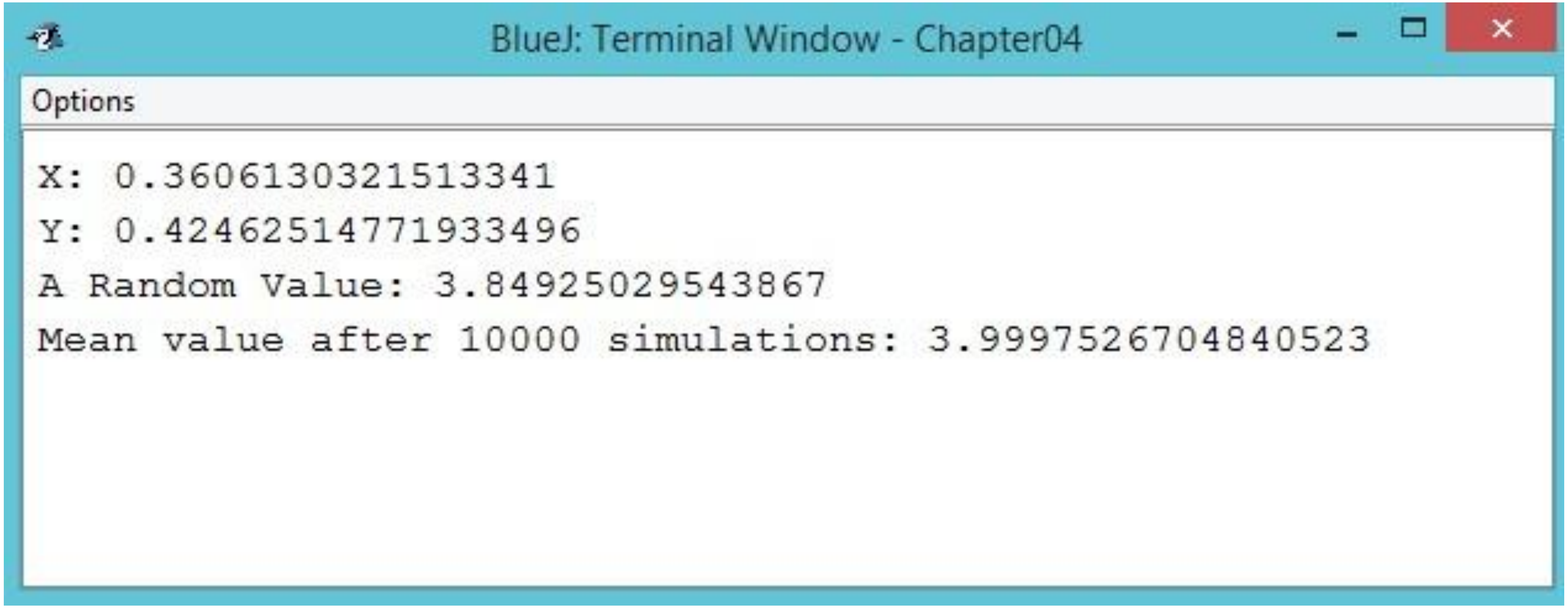


$f(x) = 4/a^2 * x$ if $0 <= x < a/2$;
$f(f) = - 4/a^2 * (x- a)$ if $a/2<= x < a$;

- This is using the same technique we used in generating a random sample over a certain range:
- **int r = (int) (Math.random() * COUNT) * STEPS + BASELINE;**

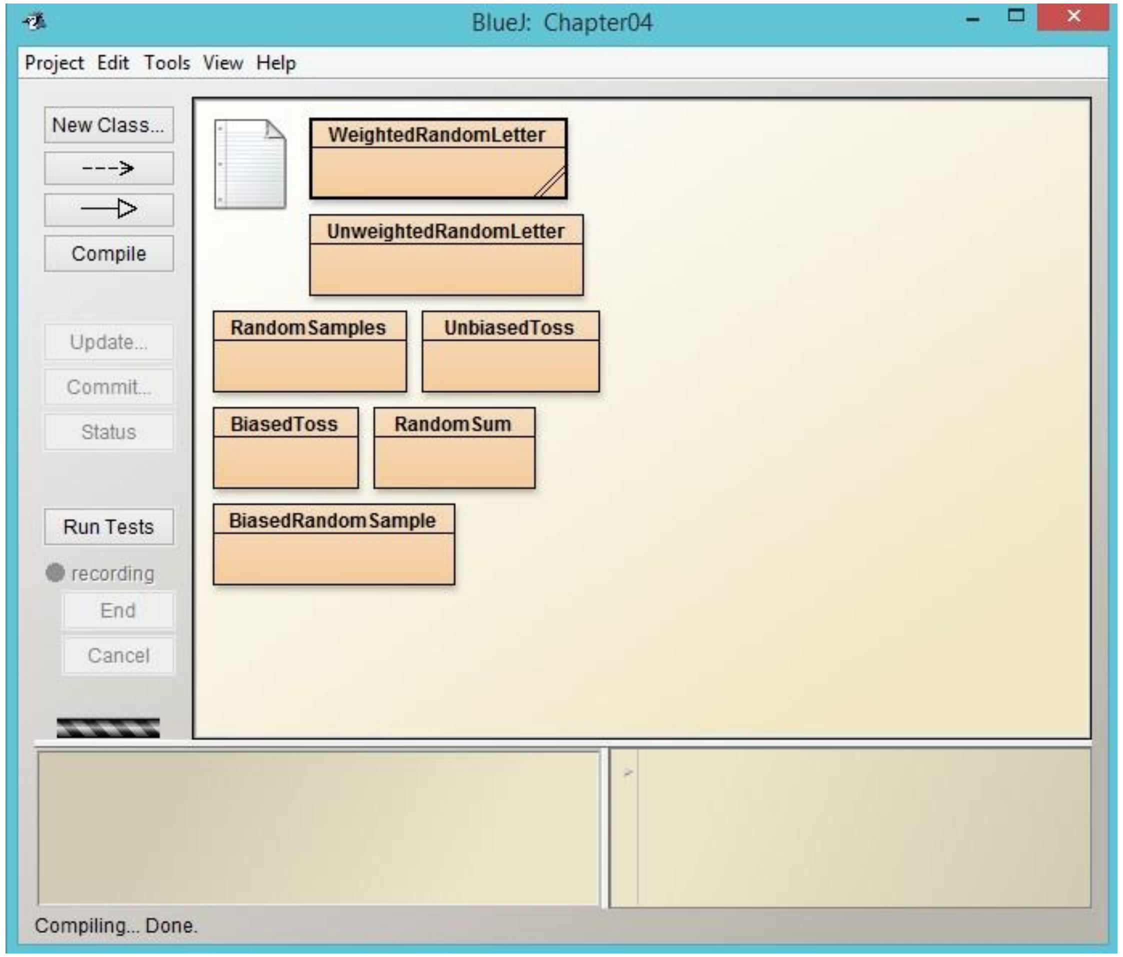
SPAN = 2.0, BASELINE = 3.0;

Run many times, the mean value is approaching BASELINE + SPAN/2.0;



```
X: 0.3606130321513341
Y: 0.42462514771933496
A Random Value: 3.84925029543867
Mean value after 10000 simulations: 3.9997526704840523
```

Try it out ! (example program in this lecture in a zip file, download and try.)



Compiling... Done.