EECS 1710
Programming for Digital Media

Lecture 7 :: Decision Making

YORK UNIVERSITÉ UNIVERSITY

Important Announcements

- Midterm
  - Next Wednesday **<u>Oct 5, 2022</u>**
    - 12:40pm – 1:20pm (40 mins)
    - Written exam
    - Worksheets & sample test will be posted this week

- Labtest 1 (after reading week)
  - October 19, 10:30am – 12pm
  - Labtest will start at 10:30am (closes at 12noon) – 1hr 30mins
  - We will do a short 30min sample test run (in lab3 – Oct 5) from 11am - 11:30am

YORK UNIVERSITÉ UNIVERSITY

This Week

Lecture 7

- Error Types
- Strings & string methods (revisited)
- text(), textWidth(), textSize()
- random()
- relational and conditional operators

Lecture 8

- *conditional logic practice*
- *Branching*
- *IF, ELSE*
- *SWITCH, CASE*

YORK UNIVERSITÉ UNIVERSITY

First, a note on errors …

In lab1, you may have encountered 3 types of error:

- SYNTAX errors
  - Compile time (i.e. statement is illegally specified, and you will be prevented from running – mostly detected by the PDE)
- RUNTIME errors
  - Program can run, but may cause an exception (e.g. a variable has an unexpected value)
  - E.g. `int y = 5 / x;` // causes div by zero exception if x=0
    output: `ArithmeticException: / by zero`
- LOGIC errors
  - runs but output is wrong (nothing to tell you in PDE, or at runtime)
  - E.g. `int area = width + height;`

YORK UNIVERSITÉ UNIVERSITY

More on Methods!

YORK UNIVERSITÉ UNIVERSITY

String Expressions

- We can use the '+' operator on Strings to join them together!

```
String strWorld = "World";
String str = "Hello" + " " + strWorld;
String str2 = str + "\n" + "EECS" + 1710;
```

- We can join any type to a string using a string expression
  - The type will be converted automatically to a string

other String methods in Processing?

```
String str = “    hello world  ”;
print(trim(str));
```

String Functions

`join()`	Combines an array of `Strings` into one `String`, each separated by the character(s) used for the `separator` parameter
`matchAll()`	This function is used to apply a regular expression to a piece of text
`match()`	The function is used to apply a regular expression to a piece of text, and return matching groups (elements found inside parentheses) as a `String` array
`nf()`	Utility function for formatting numbers into strings
`nfc()`	Utility function for formatting numbers into strings and placing appropriate commas to mark units of 1000
`nfp()`	Utility function for formatting numbers into strings
`nfs()`	Utility function for formatting numbers into strings
`splitTokens()`	The `splitTokens()` function splits a `String` at one or many character ”tokens”
`split()`	The `split()` function breaks a string into pieces using a character or string as the divider
`trim()`	Removes whitespace characters from the beginning and end of a `String`

Remove whitespace (spaces, tabs, newlines)

Formats numbers into strings (e.g. with nearest decimal places, commas, etc

Printing text/typography to app window?

- To do this, make use of these methods:

Typography

	`PFont`	Grayscale bitmap font class used by Processing
Loading & Displaying	`createFont()`	Dynamically converts a font to the format used by Processing
	`loadFont()`	Loads a font into a variable of type `PFont`
	`textFont()`	Sets the current font that will be drawn with the `text()` function
	`text()`	Draws text to the screen
Attributes	`textAlign()`	Sets the current alignment for drawing text
	`textLeading()`	Sets the spacing between lines of text in units of pixels
	`textMode()`	Sets the way text draws to the screen
	`textSize()`	Sets the current font size
	`textWidth()`	Calculates and returns the width of any character or text string
Metrics	`textAscent()`	Returns ascent of the current font at its current size
	`textDescent()`	Returns descent of the current font at its current size

YORK UNIVERSITÉ UNIVERSITY

Capturing and displaying user key presses

```
// A rough capture of key presses that append
// the characters to the end of the string
// and display using the text() method

String str = "";

void setup() {
  size(800, 600);
  stroke(255, 255, 255);
  textSize(40);
}

void draw() {
  background(0, 0, 0);
  text(str, 100, 300);
}

void keyPressed() {
  // key is the character pressed and
  // concatenated onto the current string str
  str += key;
}
```

No filters on these characters

YORK UNIVERSITÉ UNIVERSITY

Math → random()

- Very useful:

Syntax

```
random(high)
random(low, high)
```

Picks a value x between (0 <= x < high)

Picks a value x between (low <= x < high)

Parameters

low	(float)	lower limit
high	(float)	upper limit

Return

float

YORK UNIVERSITÉ UNIVERSITY

Random float or int?

- Combine some methods

```
// RANDOMLY PICK A NUMBER BETWEEN low & high (inclusive)

final float LOW = 0;
final float HIGH = 10;

void setup() {
}

void draw() {

  float value = random(LOW, HIGH+1);

  println("random real between (" + low + "," + high + ") = " + value);
  println("random int  between (" + low + "," + high + ") = " + floor(value));
}
```

Random Colours

```
// Random background colour

// random(high); or random(low,high);
final float LOW = 0;
final float HIGH = 256;

void setup() {
  size(300,300);
}

void draw() {
  // floor ensures we never pick higher than 255
  float red = floor(random(LOW, HIGH));
  float green = floor(random(LOW, HIGH));
  float blue = floor(random(LOW, HIGH));

  background(red,green,blue);
  //delay(100);
}
```

Side Note

- delay()

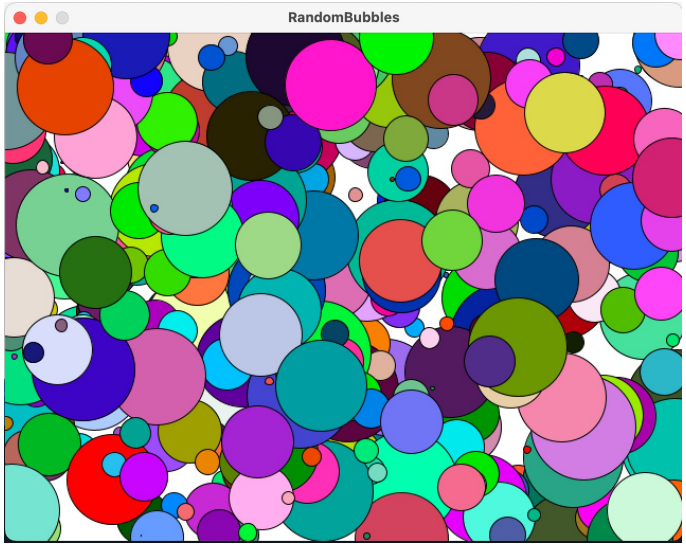
Syntax	`delay(napTime)`	
Parameters	**`napTime`** `(int)`	milliseconds to pause before running draw() again
Return	`void`	

- frameRate()
  - default is 60 fps (frames per second)
  - can set the frameRate(fps);
  - or access the system variable: println(frameRate);

YORK U
UNIVERSITÉ
UNIVERSITY

Random Objects

```
// RANDOM Circles with Random Colours

// CREATE RANDOMLY SIZED CIRCLES of RANDOM COLOURS in the APP WINDOW


float posX;
float posY;
float radius;

void setup() {
  size(640, 480);
  background(255,255,255);
}

void draw() {
  posX = random(0,width);
  posY = random(0,height);
  radius = random(100);

  fill( floor(random(0,256)), floor(random(0,256)), floor(random(0,256)) );
  circle(posX, posY, radius);

  //delay(1000);
}
```



Making Decisions [1]

Relational & Conditional Operators

YORK UNIVERSITÉ UNIVERSITY

Relational Operators

numeric operands, boolean result

Precedence	Operator	Operands	Syntax	true if
-7 ➔	<	numeric	x < y	x is less than y
	<=	numeric	x <= y	x is less than or equal to y
	>	numeric	x > y	x is greater than y
	>=	numeric	x >= y	x is greater than or equal to y
-8 ➔	==	any type	x == y	x is equal to y
	!=	any type	x != y	x is not equal to y

```
double ratePercent = 33.34;
boolean isLTE50 = (ratePercent <= 50.0);
```

YORK UNIVERSITÉ UNIVERSITY

Boolean Expressions

- Using relational/conditional operators, we can construct boolean expressions (much like we did with numeric expressions)

- Again, note: the RHS of the expression is resolved completely before assignment to a result variable

- Examples:

```
double ratePercent = 3.84;
boolean isUnusual = ratePercent < 2.0;
boolean isRisky = ratePercent*4 > 8.0 ;
```

YORK UNIVERSITÉ UNIVERSITY

Conditional operators:

boolean operands, boolean result

Condition	Operator	Example

If one condition AND another condition	&&	int i = 2;
		int j = 8;
		((i < 1) && (j > 6))

If either one condition OR another condition	||	int i = 2;
		int j = 8;
		((i < 1) || (j >= 10))

NOT	!	int i = 2; (!(i > 3))

```
double ratePercent = 3.84;

boolean isUnusual = ratePercent<2.0 || ratePercent>8.0 ;

boolean isValid = ((ratePercent > 0) && (ratePercent < 100));
```

```
double ratePercent = 3.84;

boolean isUnusual = ratePercent<2.0 || ratePercent>8.0 ;

                                false || ratePercent>8.0 ;

                                false || false ;

            isUnusual =             false ;
```

```
double ratePercent = 3.84;

boolean isValid = ratePercent>0 && ratePercent<100 ;

                  true && ratePercent<100 ;

                  true && true ;

        isValid =       true ;
```

Truth table (conditional operators)

a	b	a && b	a || b	!a	!(a||b)
false	false	false	false	true	true
true	false	false	true	false	false
false	true	false	true	true	false
true	true	true	true	false	false

Examples

- Method to test if a pixel is a valid colour
- Recall → each colour channel should be between 0-255

```
boolean isValidPixel(int r, int g, int b) { … }

isValidPixel(150,58,33)    → true
isValidPixel(150,358,33)   → false
isValidPixel(-1,-100,256)  → false
```

isValidPixel()

```
// isValidPixel()

boolean isValidColour(int col) {
  boolean valid = (col>=0) && (col<=255);
  return valid;
}

boolean isValidPixel(int r, int g, int b) {
  return (isValidColour(r) && isValidColour(g) && isValidColour(b));
}

void setup() {
  println("isValidPixel():");
  println("(150,58,33) -> " + isValidPixel(150, 58, 33));
  println("(150,358,33) -> " + isValidPixel(150, 358, 33));
  println("(-1,-100,256) -> " + isValidPixel(-1, -100, 256));
  println("(255,255,255) -> " + isValidPixel(255, 255, 255));
}

void draw() {
}
```

Examples

- Method to test if a position (x,y) is inside the application window

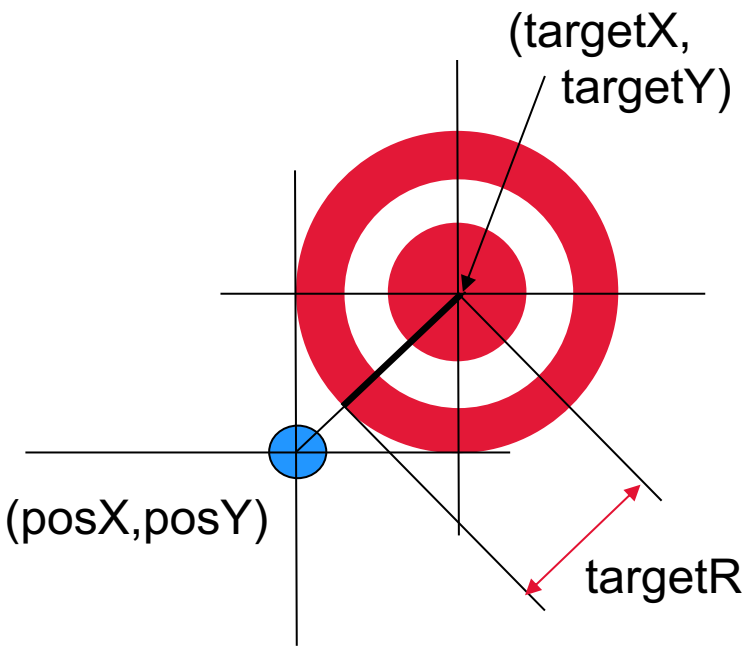
```
boolean isWithin(float posX, float posY) { … }


// assume width=640, height=480

isWithin(-1,0)       → false
isWithin(641,480)    → false
isWithin(1,1)        → true
isWithin(600,400)    → true
```

isWithin()

```
// isWithin()

// tests to see if a point is inside the app window
boolean isWithin(float posX, float posY) {
  boolean isInsideX = (posX>=0 && posX<=width);
  boolean isInsideY = (posY>=0 && posY<=height);
  return (isInsideX && isInsideY);
}

// assume width=640, height=480
void setup() {
  size(640,480);

  println("isWithin(-1,0)    -> " + isWithin(-1,0));
  println("isWithin(641,480) -> " + isWithin(641,480));
  println("isWithin(1,1)     -> " + isWithin(1,1));
  println("isWithin(600,400) -> " + isWithin(600,400));
}

void draw() {
}
```

projectile motion with target

```
void generateTarget() {
  targetX = random(width/4,3*width/4);
  targetY = random(height/4, 3*height/4);
  targetR = random(100);
}

void keyPressed() {
 generateTarget();
}
void drawTarget() {
  stroke(0,0,0);
  fill(255,0,0);
  circle(targetX,targetY,targetR);
  fill(255,255,255);
  circle(targetX,targetY,2*targetR/3);
  fill(255,0,0);
  circle(targetX,targetY,targetR/3);
}

boolean hitTarget(float posX, float posY) {
 return ( sqrt(pow(posX-targetX,2)+pow(posY-targetY,2)) <= targetR );

}
```



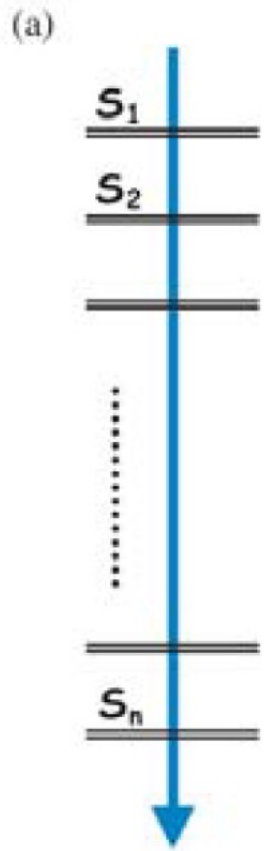
Making Decisions [2]

Branching

YORK UNIVERSITÉ UNIVERSITY

Flow of execution / Flow of control

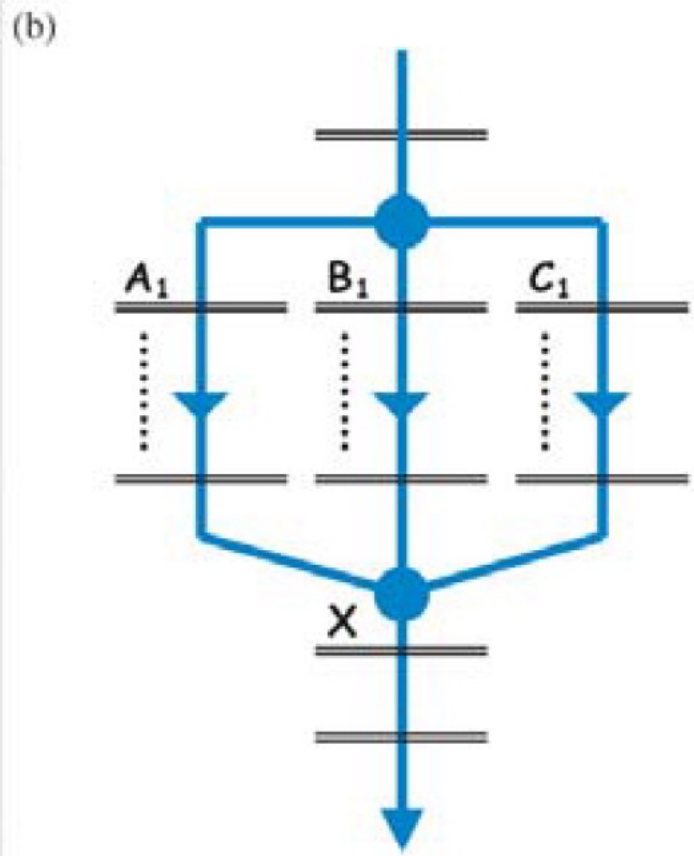
- **Sequential** vs. Branching




```
// … some statements …


// S1
{ … }
// S2
{ … }
// S3
{ … }

…
// S1

// … more statements …
```

Flow of execution / Flow of control

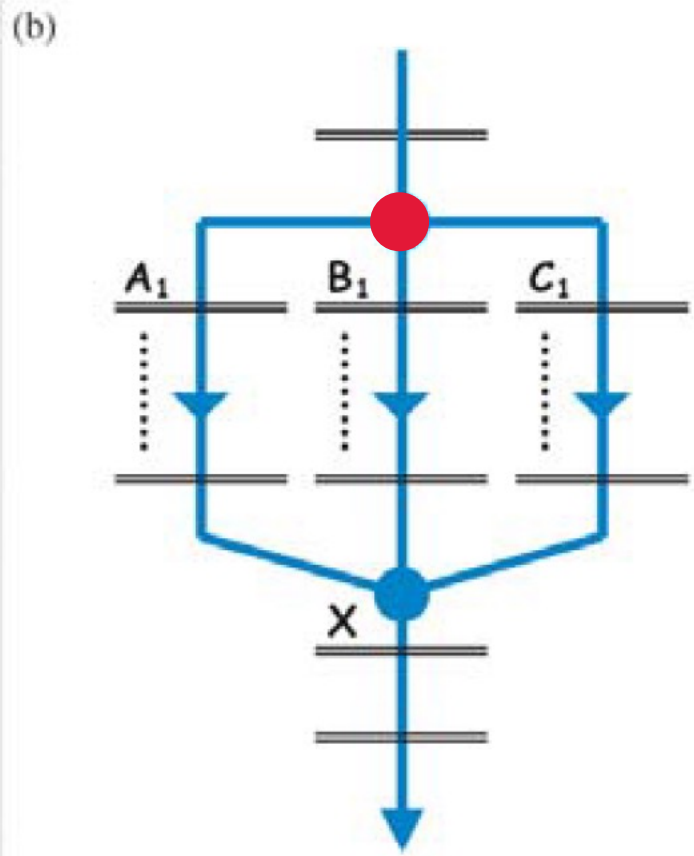
- Sequential vs. **Branching**




```
// … some statements …

// perform some test

// execute if test gives outcome A1
{ … }
// execute if test gives outcome B1
{ … }
// execute if test gives outcome C1
{ … }

// … more statements …
```

Flow of execution / Flow of control

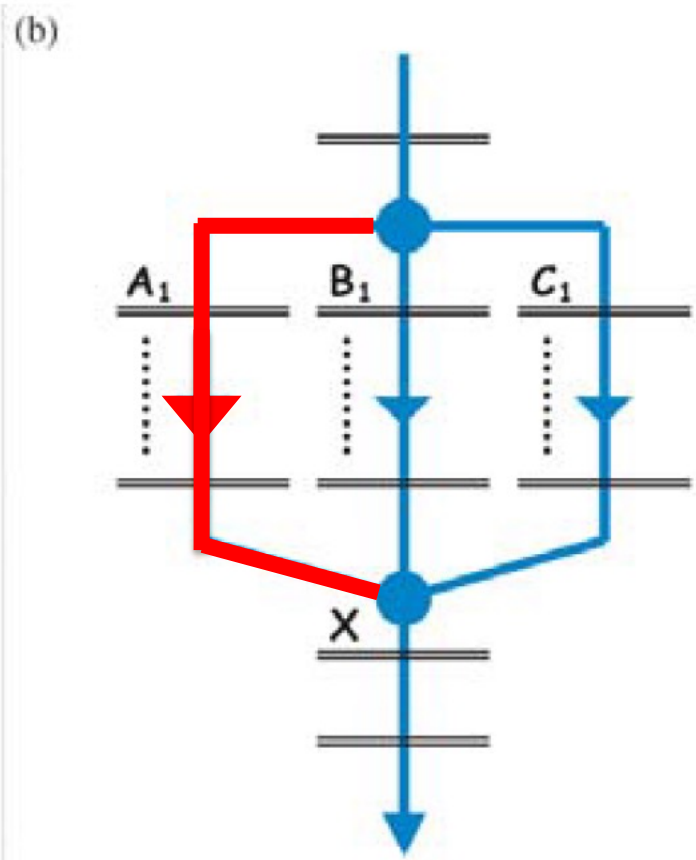
- Sequential vs. **Branching**




```
// … some statements …

// perform some test

// execute if test gives outcome A1
{ … }
// execute if test gives outcome B1
{ … }
// execute if test gives outcome C1
{ … }

// … more statements …
```

Flow of execution / Flow of control

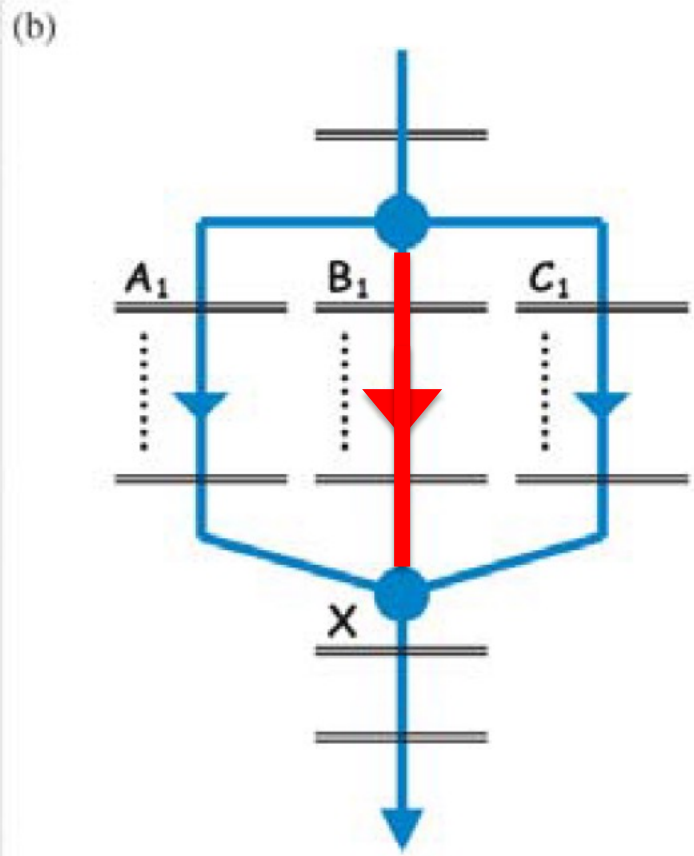
- Sequential vs. **Branching**




```
// … some statements …

// perform some test

// execute if test gives outcome A1
{ … }
// execute if test gives outcome B1
{ … }
// execute if test gives outcome C1
{ … }

// … more statements …
```

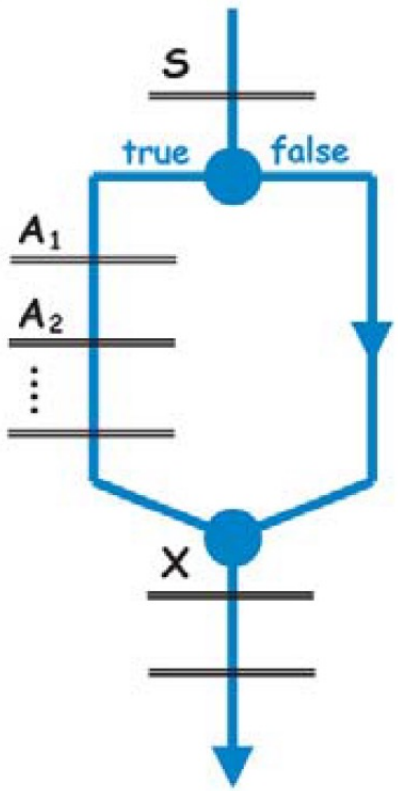
Flow of execution / Flow of control

- Sequential vs. **Branching**

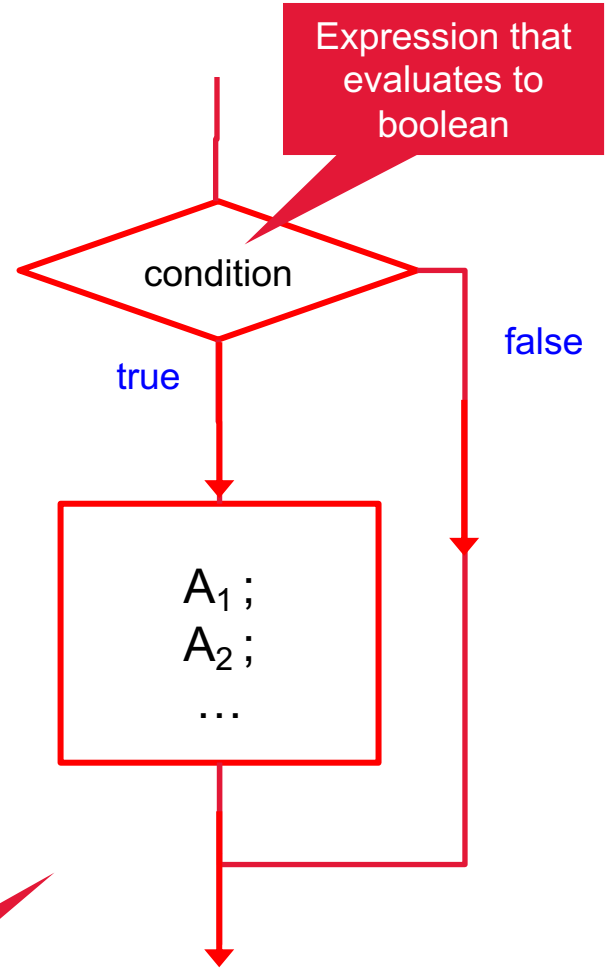



```
// … some statements …

// perform some test

// execute if test gives outcome A1
{ … }
// execute if test gives outcome B1
{ … }
// execute if test gives outcome C1
{ … }

// … more statements …
```

The IF statement




```
Statement-S
if (condition)
{
    Statement-A1
    Statement-A2
        .
        .
        .
        .
}
Statement-X
    .
    .
    .
    .
```



flowchart

keyPressed

- Built-in boolean variable (not the method)

```
void draw() {

  if (keyPressed) {

     println("Key has been pressed!");
     println("Key was: " + key);

     if (key == 't') {
         generateTarget();
     }

  }

}
```

Hit test for projectile

```
// added global variable
boolean hit = false;


// inside moveProjectile(), use x,y with hit
test
if (hitTarget(x,y)) {
  hit = true;
}

void draw() {
  t+=0.05;
  if (!hit) moveProjectile(x0,y0,v0,theta,t);
  drawTarget();
}

void keyPressed() {
 generateTarget();
 hit = false;
}
```