Outline

• Turtle
  • Looping
  • while Loops
  • for Loops
  • Choosing the Right Loops

Introduction to Turtle (1/2)

• Before we see loops, we need some tools
  • We will use a Turtle 🐢 to help us understand loops
  • Turtles are based on Seymour Papert's Logo*, a language for beginners
  • Turtles 🐢 are imaginary pens that when given instructions can draw shapes for us

Introduction to Turtle (2/2)

- Turtles know where they are, what direction they are facing, and how to move and turn.
- Turtles can draw lines behind them as they move around the screen or just move without drawing.
- PaneOrganizer holds instructions for the turtle – reminiscent of our first Robot example...

Turtle’s Methods (1 of 2)

TAs have written a Turtle class

```java
public class Turtle {
    // instance variables elided
    /* constructor for Turtle instantiates a Polygon representing the Turtle graphically */
    public Turtle() {
        // some code here
    }
    /* reset turtle to center of pane */
    public void home() {
        // some code here
    }
    /* turn right a specified number of degrees */
    public void right(double degrees) {
        // some code here
    }
    /* turn left a specified number of degrees */
    public void left(double degrees) {
        // some code here
    }
    // continued

    /* move forward a specified distance, drawing a line as the turtle moves */
    public void forward(int distance) {
        // some code here
    }
    /* move backward a specified distance, drawing a line as the turtle moves */
    public void back(int distance) {
        // some code here
    }
    /* move turtle to a specified position without drawing a line */
    public void setLocation(Point2D loc) {
        // some code here
    }
    /* return turtle’s location */
    public Point2D getLocation() {
        // some code here
    }
    /* return the Polygon (the triangle) contained in Turtle class so that we can graphically add it in the P.O. */
    public Shape getShape() {
        // some code here
    }
}
```

Turtle’s Methods (2 of 2)
Drawing with Turtle (1/2)

- Need class to tell Turtle how to draw some basic shapes
  - Will contain a Pane and a Turtle
  - Add these methods for each shape we want to draw
- First, determine what shapes we want
  - This lecture: square, random walk

Drawing with Turtle (2/2)

- How will we code it?
  - Create PaneOrganizer class which defines methods for drawing each shape
  - PaneOrganizer also instantiates the root Pane that the Turtle will draw on and contains the Turtle. The root Pane is a wrapper class that contains a polygon and defines methods for how the Turtle will move.
  - It can also return its polygon as a node via getShape()

A Repetitive Solution (1/2)

- Let's write drawSquare method in the PaneOrganizer class
- Brute force: write line of code for each side of the square
What if we wanted to make a more general method that handles regular shapes such as pentagons or octagons?

- Need to call `forward()` and `right()` for each side.
- Cannot determine in advance how many sides we need in generic method.
- Note that we’re using the Turtle’s primitive methods to generate higher-level shapes that are normally already defined in JavaFX.

There must be an easier way!

Outline

- Turtle
  - Looping
    - `while` Loops
    - `for` Loops
    - Choosing the Right Loops

Looping (1/2)

- Execute a section of code repeatedly.
  - Use `boolean` (true and false) as loop conditions; continues looping as long as condition is true, but when `boolean` is false, loop condition equals exit condition and loop is terminated.
  - As with conditionals, code in loop can be a single line or many lines enclosed in curly braces.
  - Section of code executed is called loop’s body.
Looping (2/2)
- Three loop structures in Java
  - while loop
  - do while loop
  - for loop
- Differ in relation between body and loop condition, as well as length of execution
- Let's look at while loop first

Outline
- Turtle
- Looping
  - while Loops
  - for Loops
  - Choosing the Right Loops

The while loop (1/2)
- Executes while stated condition is true
  - tests loop condition before executing body
  - if loop condition is \textit{false} first time through, body is not executed at all
  
  ```java
  while (loop condition) {
      loop body;
  }
  ```
The **while** loop (2/2)

- **Examples of loop conditions:**
  - `numClasses < 6`
  - `peopleStanding <= maxPeople`
  - `this.checkAmount() <= acctBalance`
  - `this.isSquare()`  
    // predicate, a method that returns a boolean

- Follows the same rules as conditions for **if-else** statements
- Multiple conditions can be combined using logical operators
  - `(and (&&), or (||), not (!))`
    
    - `(numClasses >= 3) && (numClasses <= 5)`
    - `(peopleStanding <= maxPeople) || (maxPeople < 50)`

---

### **while** loop Flowchart (1/2)

- **while** loops continue **while** the loop condition is true
- `<loop condition>` can be any Boolean expression

---

### **while** loop Flowchart (2/2)

- **while** loops continue **while** the loop condition is true
- `<loop condition>` can be any Boolean expression
All Flow of Control Structures: 1-in, 1-out

- Benefits of predictable flow of control
  - Much easier debugging
  - Compiler can optimize much better
- Different from “spaghetti” code (unorganized and difficult to maintain code) with goto methods to allow program to jump to another line of code
  - Go To Statement Considered Harmful letter by Edsger Dijkstra, CACM, 1968
  - If-else, etc., are “structured flow-of-control”

So, just how bad is goto?

Syntax: Random Walk Using while

- Method in PaneOrganizer class:
  - draws random lines while this.turtle is within its pane
  - On last step of walk, turtle will move forward out of pane
    - The line is clipped by JavaFX since we don’t explicitly tell it to wrap around
    - No point in continuing to walk outside the pane

```java
public void randomWalk() {
    // while this.turtle's position is inside its pane, move this.turtle randomly
    // this.turtle's initial location set to (0,0)
    // this.turtle's final location set to (x, y) = (192, 120)
    while (this.root.contains(this.turtle.getLocation())) {
        this.turtle.forward((int) (Math.random() * 15));
        this.turtle.right((int) (Math.random() * 360));
    }
}
```
TopHat Question 1
What is the value of tempSum after this while loop is terminated?

```
int tempSum = 0;
while(tempSum < 10) {
    tempSum += 3;
}
```

A. 10
B. 9
C. 12
D. The loop will never terminate

The do while Loop
- do while always executes loop body at least once by switching order of test and body
- <loop condition> is Boolean expression

Example: Another Random Walk
- Method of PaneOrganizer class:
  - draws random lines while turtle is within pane
  - this.turtle starts in center of root pane, so first step guaranteed to be within pane

```
public void centeredRandomWalk() {
    // moves turtle to pane's center
    this.turtle.home();
    // moves turtle randomly within pane
    do {
        this.turtle.forward((int)(Math.random()*15));
        this.turtle.right((int)(Math.random()*360));
    } while (this.root.contains(this.turtle.getLocation()));
}
```
do while vs. while (1/2)

- In both loops:
  - stops executing body if loop condition is false
  - must make sure loop condition becomes false by some computations to avoid an “infinite loop”
  - infinite loop means your loop condition will never turn false – i.e., exit condition never occurs (and your program “freezes up!”)

- do while:
  - body always executes at least once
  - loop condition tested at bottom of loop body

- while:
  - body may not execute at all
  - loop condition tested before body
  - loop condition variables must be set before loop entry
  - useful for screening bad data that might cause statements within loop to fail (e.g. while (ref != null))

TopHat Question 2

What’s the difference between these two loops?

Loop 1:
while(andyIsAway()) {
  this.tas.takeADayOff();
}

Loop 2:
do {
  this.tas.takeADayOff();
} while (andyIsAway());

A. In the second loop, the condition is tested before the body
B. In the second loop, the TAs always take at least 1 day off
C. In the first loop, the body is executed before the condition is tested.
D. There is no difference between the two loops
Outline
• Turtle
• Looping
• while Loops
• for Loops
• Choosing the Right Loops

for loops (1/4)
● Most specialized loop construct (and the first high-level, goto-less loop in FORTRAN); typically used to execute loop body a predetermined number of times
  ○ while and do while loops can execute body for undetermined number of times; based on boolean
● This is the syntax for a for loop:
  ```javascript
  for (<init-expr>; <loop condition>; <update>) {
    <loop body>
  }
  ```

for loops (2/4)
```javascript
for (<init-expr>; <loop condition>; <update>) {
  <loop body>
}
```
for loops (3/4)

for (<init-exp>; <loop condition>; <update>) {
    <loop body>
}

● <loop condition>
    ○ true or false
    ○ test involves loop counter to determine if loop should execute
    (e.g. \( i < 5 \))
    ○ checked at start of every loop (including the first)

for loops (4/4)

for (<init-exp>; <loop condition>; <update>) {
    <loop body>
}

● <update>
    ○ expression that modifies loop counter
    ○ executed at end of every <loop body>, just before returning to the top of the loop
    ○ (e.g. \( i++ \)) this would increase the loop counter by 1 each loop

drawSquare Revisited

Better way of drawing square rather than explicitly drawing each side:

```java
public void drawSquare(int sideLen) {
    /* start with integer i initialized to 0; execute as long as i < 4; each execution increments i by 1 at the bottom of the loop */
    for (int i = 0; i < 4; i++) {
        this.turtle.forward(sideLen);
        this.turtle.right(90);
    }
}
```
for Flowchart

- for loop has four parts
  - initialize value of counter
  - test loop condition
  - loop body
  - update counter

---

for Flowchart

- We can use an example of a student reading books on different floors of the SciLi.

  ```java
  Student student = new Student("Sarah");
  student.goToSciLi();
  for (int floor = 1; floor < 15; floor++) {
    student.readBook(); //read a new book
  }
  student.goHome();
  ```

---

Outline

- Turtle
- Looping
  - while Loops
  - for Loops
- Choosing the Right Loops
Choosing the Right Loop (1/2)

- **for** loop is called a **definite** loop because you can typically predict how many times it will loop.
- **while** and **do while** loops are **indefinite** loops, as you do not know when they will end.
- **for** loop is typically used for math-related loops like counting finite sums and sequentially looping through elements of an array (Thursday's Lecture).

Choosing the Right Loop (2/2)

- **while** loop is good for situations where **boolean** condition could turn **false** at any time.
- **do while** loop is used in same type of situation as **while** loop, but when code should execute at least once.
- When more than one type of loop will solve problem, use the cleanest, simplest one.

TopHat Question 3

What is the value of **sum** at the end of the following loop?

```
sum = 0;
for (int i = 0; i <= 10; i+=2) {
    sum++;
}
```

A. 10    B. 11    C. 5    D. 6
Syntax: Nested Loops
● Loops, just like if statements, can be nested!
● Example: `drawFilledSquare`

```java
public void drawFilledSquare(int sideLen) {
    // fill in concentric squares
    for (int i = 0; i < (sideLen/2); i++) {
        for (int j = 0; j < 4; j++) {
            this.turtle.forward(sideLen - (2*i));
            this.turtle.right(90);
        }
        /* note we can use loop counter R/O (read-only) in body but never reset it there! */
        // position turtle for next iteration
        this.turtle.right(90);
        this.turtle.forward(1);
        this.turtle.left(90);
        this.turtle.forward(1);
    }
}
```

● What does this do?
  o decrementing `sideLen` by 2 each iteration to guarantee that each “inner square” drawn in the inner loop is exactly one unit away on either side from square immediately “outside” of it hence, one + one = two)

Syntax for Nested Loops Explained
● Turtle is represented by
● What is the outer loop doing?
  o first draws outer square
    ▶️ Turtle starts upright!
    ▶️ Move forward 1 unit!
    ▶️ Rotate 90 degrees right!
    ▶️ Draw inner square
  ▶️ Position turtle for each iteration
    ▶️ Move forward 1 unit!
    ▶️ Rotate 90 degrees left!
    ▶️ Move forward 1 unit!
  ▶️ Draw inner square
    drawFilledSquare draws concentric squares; each individual square is drawn using the nested loop

Looping to Make a Filled-in Design(1/2)
● 3D Printing Food!!
Decrementing Counter

- We can count backwards in our loop too
  - just change the counter update expression
  - in fact, we can update however we want

- for loops end in one of two ways
  - when counter value equals limit (for < or >)
  - when counter value "goes past" limit (for <= or >=)

- thus, countDownSeconds() would also print 0 if used i >= 0
- beware of such "off-by-one" errors! — hand simulation really helps!

```java
public void countDownSeconds(){
    /* change counter to decrement, and change the loop condition accordingly */
    for(int i = 5; i > 0; i--){
        System.out.print(i);
    }
}
```

break

- break causes immediate exit from a flow-of-control structure (e.g., switch, while, do while, for)
- Example:

```java
for (int i = 0; i < 5; i++){
    if (this.cookieJar.getNumberOfCookies() == 0) {
        break;
        // If there are no cookies left, we should break out of the loop!
    }
    this.eatACookie();
}
// Execution continues here after loop is done or after break statement is executed
```

- Execution continues with first line of code after structure
- There are other ways to do this loop...


**continue**

- When used in `while`, `for`, or `do while` structures, `continue` skips remaining statements in body of that structure and proceeds with next iteration of loop.
  - It is useful if there is a list of data that you are looping over and you want to skip processing of data that is somehow “not legal”.
- In `while` and `do while` structures, execution continues by evaluating loop-continuation condition.
- In `for` structure, execution continues by incrementing counter and then evaluating loop condition.

**continue Example**

```java
/* We oversee letting kids on a rollercoaster ride if they are tall enough */
for (int i = 0; i < 20; i++) {
    if (!ride.kidIsTallEnough(i)) {
        // if the kid at i is not tall enough
        // skip to the next iteration (the next kid in line)
        continue;
    }
    this.rideRollercoaster(ride.getKid(i)); // let kid onto ride
}
// more code here
```

**Boolean Predicates and Flags**

- A **Boolean predicate** is a method that returns a `boolean` (e.g., `isLeft()`, `isAvailable()`, `kidIsTallEnough()`).
- A **Boolean flag** records the result of a predicate; set and saved in one place, used later in different place.

**Example (implementing a `for` loop, using `while`):**

```java
boolean isDone = false;
int i = 0;
while (!isDone) {
    i++;
    if (i == 5) {
        isDone = true;
    }
}
```

Note: Here, the Boolean flag is set within the loop, which is not legal. It is not practical.
TopHat Question 4

In the loop to the right, what is the value of `i` upon exit?

A. 4  
B. 5  
C. 6  
D. Infinite loop  

```java
boolean isDone = false;
int i = 0;
while (!isDone){
    i++;
    if(i == 5){
        isDone = true;
    }
}
```

Empty Intervals

- Example scenario: we want to keep a running sum of a sequence of numbers
- What happens if we try to add integers in this loop?

```java
public int sum() {
    int tempSum = 0;
    for (int i = 1; i < 1; i++) {
        tempSum += i;
    }
    return tempSum;
}
```

- Answer: body of loop is not executed
- Why?
  - loop condition is `false` for initial counter value

Correct Example

- What about this loop?

```java
/*This method sums all numbers from 1 up to and including 10*/
public int sum() {
    int tempSum = 0;
    for (int i = 1; i <= 10; i++) {
        tempSum += i;
    }
    return tempSum;
}
```

- It will work!
Off-by-one Errors

- These errors occur when loop executes one too many or one too few times
  - example: add even integers from 2 to some number, inclusive
    ```java
    count = 2;
    result = 0;
    while (count < number) {
        result += count;
        count += 2;
    }
    ```
  - Should be:
    ```java
    while (count <= number) {
        // loop body elided
    }
    ```
  - Produces incorrect result if `number` is assigned an even value. Values from 2 to `number-2` will be added (i.e., `number` is excluded)

Syntax: Other Loop Errors (1/2)

- Make sure test variables have proper values before loop is entered
  ```java
  // example 1
  int product = 0;
  while (product < 100) {
      product *= 2;
  }
  ```
- Make sure tests check proper conditions
  ```java
  // example 2
  int product = 0;
  while (product < 100) {
      product *= 2;
  }
  ```

TopHat Question 5

Given the following code:

```java
int num = 2023;
while (num > 0) {
    num--;
} while (num < 2023);
```

What do you expect will happen?

A. Loop will never end
B. Loop will run 2023 times (until `num` is 0), then end
C. Loop will run only once
Syntax: Other Loop Errors (2/2)

- ALWAYS HAND SIMULATE first, last, and typical cases in a loop to avoid off-by-one or infinite loop errors
  - the first and last cases of a loop's execution are called boundary conditions or edge cases or corner cases
  - hand simulation doesn't just apply to loops — use it for everything!
  - Trust us — it saves debugging time!

Which loop to use?

- You want to stack 17 blocks
- Your job is to stand at the end of the bowling alley and pick up all the pins, one by one, that have been knocked over
- Sleep until your clock reads 7:51AM or later

Announcements

- Collaboration Policy Phase 2 Quiz
- Cartoon Deadlines
  - Early due Thursday 10/19
  - On-time due Saturday 10/21
  - Late due Monday 10/23
  - Lab 5 — GitHub and Debugging this week
  - Don't forget to wash Cartoon Mini-Assignment
  - Tshirt form is out now and open until Thursday 10/19
Privacy and Surveillance I: A Brief History

Past Week’s SRC Lab!
Google Personal Data Insights

Source: adssettings.google.com
Privacy: A Hot Topic Today!

From 2019 Study
Source: Global Data Privacy Laws (Greenleaf 2019)

Note: Having a data privacy law is not the same as enforcing it properly!

From 2023 Update to Study
Source: Global Data Privacy Laws (Greenleaf 2023)
A BRIEF HISTORY: Analog Communication

Before...

- Rolodex
- Answering Machine
- VHS tapes
- FAX machines
- Portable CD player/Walkman
- Separate mouthpiece phone
- Princess phone
- Telephone


The Digital Shift

Now:

- Image sources: Apple, Spotify, Google

"There are only two industries that call their customers 'users': illegal drugs and software" (Edward Tufte)

COMMON ENCOUNTERS

Image sources: Google, Intego, Apple, Meta, Mozilla, Hallakate

Privacy
A BRIEF HISTORY

Early American Surveillance

- FBI mission: “Protect American people and uphold the Constitution of the United States”
- Investigate criminal enterprise and domestic terrorism, subversion of the state etc.

J. Edgar Hoover testifying before the House on Un-American Activities Committee (1947)

Photo Credit: biography.com

FBI: domestic cases, investigate crimes, work w/ Dept of Justice

1908

1947

1952

CIA: foreign intelligence to policymakers for national security decisions, prohibited from collecting info on US persons

NSA: counterintelligence against adversaries and foreign intelligence

1978: Foreign Intelligence Surveillance Act
- US Federal Law: physical + electronic surveillance
- FISC: oversee requests for surveillance warrants by federal law enforcement + intelligence agencies

2007: Protect America Act
- Electronic communications monitored by NSA (previously restricted to non-Americans)
- No more absolute privacy
- Without court order/ oversight

Image credits: Wikimedia Commons, Tech Xplore
Edward Snowden/ NSA + CIA
A BRIEF HISTORY: Whistleblowers

PRISM: Top-secret NSA program leaked by Snowden
- NSA can 'reach directly into servers' from Facebook, Apple, Microsoft, Google, Twitter and other platforms
- Read communications + real-time collection (chat, videos, photos, file transfers)
- Critique: did 'grave damage' to American intelligence capacities
- Violated 'Espionage Act of 1917' + theft of government property
- US Passport revoked

Recent news on citizenship:

Photo credits: New York Times, Electronic Frontier Foundation, Reuters

Employee of CIA & later NSA subcontractor

Edward Snowden's Book: Permanent Record

WikiLeaks
- Nonprofit organization that publishes news leaks/classified media by anonymous sources
- Founder: Julian Assange, Australian editor
- 2010: Chelsea Manning leaks documents about the Afghan + Iraq war
- 2016: Presidential election campaign
- 2016: Released thousands of pages from the DNC
- 2016: Published 20,000+ pages of emails from John Podesta, Clinton's campaign chair
- 2017-2020: Julian Assange in U.K. jail and faces potential extradition to the U.S.

Criticized for violating the personal privacy of individuals

Photo credits: The Washington Post, WikiLeaks, Reuters, The Intercept

Source: Pew Research Center

"People are more accepting of personal data collection for specific purposes."
Privacy as a Spectrum

Personal Perspectives

Source: Center for International Media Assistance

10/17/23