

public/private, While Loops

October 30, 2008

1

public vs. private

- **public** - can be used in classes other than the one in which it is declared.
- **private** - can only be used within the class in which it is declared

2

public vs. private

- **public** – can be used in classes other than the one in which it is declared.

```
public class Train {
    public void move (double dx) {
        ...
    }
}

public class TrainController {
    private Train train;
    ...
    public void onMouseDrag (Location point) {
        ...
        train.move (distance);
    }
}
```

3

public vs. private

- **private** – can only be used within the class in which it is declared

```
public class Train {
    public void move (double dx) {
        ...
    }
}

public class TrainController {
    private Train train;
    ...
    public void onMouseDrag (Location point) {
        ...
        train.move (distance);
    }
}
```

4

Producing Smoke

- How should the train produce a puff of smoke?
- What should the smoke look like?
- Where should it appear?

```
public void produceSmoke(...) {  
    ...  
}
```

5

Producing Smoke

```
public class Train {  
    ...  
    private FramedRect smokeStack;  
  
    public Train (... , DrawingCanvas trainCanvas) {  
        ...  
        smokeStack = new FramedRect (... , trainCanvas);  
    }  
  
    public void produceSmoke () {  
        FilledOval smoke = new FilledOval (  
            smokeStack.getX() + (SMOKESTACK_WIDTH - SMOKE_WIDTH) / 2,  
            smokeStack.getY() - 1.5 * SMOKE_HEIGHT,  
            SMOKE_WIDTH, SMOKE_HEIGHT, trainCanvas);  
        smoke.setColor(SMOKE_COLOR);  
    }  
}
```

Where are the
variables used in
produceSmoke
declared?

6

Producing Smoke

```
public class Train {
    ...
    private FramedRect smokeStack;

    public Train (... , DrawingCanvas trainCanvas) {
        ...
        smokeStack = new FramedRect (... , trainCanvas);
    }

    public void produceSmoke () {
        FilledOval smoke = new FilledOval (
            smokeStack.getX() + (SMOKESTACK_WIDTH - SMOKE_WIDTH) / 2,
            smokeStack.getY() - 1.5 * SMOKE_HEIGHT,
            SMOKE_WIDTH, SMOKE_HEIGHT, trainCanvas);
        smoke.setColor(SMOKE_COLOR);
    }
}
```

7

Producing Smoke

```
public class Train {
    ...
    private FramedRect smokeStack;

    public Train (... , DrawingCanvas trainCanvas) {
        ...
        smokeStack = new FramedRect (... , trainCanvas);
    }

    public void produceSmoke () {
        FilledOval smoke = new FilledOval (
            smokeStack.getX() + (SMOKESTACK_WIDTH - SMOKE_WIDTH) / 2,
            smokeStack.getY() - 1.5 * SMOKE_HEIGHT,
            SMOKE_WIDTH, SMOKE_HEIGHT, trainCanvas);
        smoke.setColor(SMOKE_COLOR);
    }
}
```

8

Producing Smoke

```
public class Train {
    ...
    private FramedRect smokeStack;

    public Train (... , DrawingCanvas trainCanvas) {
        ...
        smokeStack = new FramedRect (... , trainCanvas);
    }

    public void produceSmoke () {
        FilledOval smoke = new FilledOval (
            smokestack.getX() + (SMOKESTACK_WIDTH - SMOKE_WIDTH) / 2,
            smokestack.getY() - 1.5 * SMOKE_HEIGHT,
            SMOKE_WIDTH, SMOKE_HEIGHT, trainCanvas);    ???
        smoke.setColor(SMOKE_COLOR);
    }
}
```

9

Producing Smoke

```
public class Train {
    ...
    private FramedRect smokeStack;

    public Train (... , DrawingCanvas trainCanvas) {
        ...
        smokeStack = new FramedRect (... , trainCanvas);
    }

    public void produceSmoke () {
        FilledOval smoke = new FilledOval (
            smokestack.getX() + (SMOKESTACK_WIDTH - SMOKE_WIDTH) / 2,
            smokestack.getY() - 1.5 * SMOKE_HEIGHT,
            SMOKE_WIDTH, SMOKE_HEIGHT, trainCanvas);    ???
        smoke.setColor(SMOKE_COLOR);
    }
}
```

10

More on Constructors

- Purpose of constructors:
 - Put something on the display (by creating FilledOvals for example)
 - Remembering things created for later use (like the canvas so that we can create new shapes in other methods)

```
private DrawingCanvas smokeCanvas;

public Train (double left, double trackHeight, DrawingCanvas trainCanvas) {
    car = new FilledRect (left, top, CAR_WIDTH, CAR_HEIGHT, trainCanvas);
    ...
    smokeCanvas = trainCanvas;
}

public void produceSmoke () {
    FilledOval smoke = new FilledOval (... , smokeCanvas);
    ...
}
```

11

Midterm

- Thursday, November 6 in class, closed book, closed notes, but I will provide a cheat sheet
- Tuesday, November 4, Review in class - come with questions!
- Covers only Java, material through Scope (Oct. 28 + public/private of today)
- Types of questions, sample on line:
 - What does it do? - like our mystery programs
 - Vocabulary - I provide a program and ask you to identify all parameters, all methods, etc.
 - Find a bug - I provide a program and ask you to find the bug in it.
 - Declare variables - I provide a program and ask you to insert appropriate variable declarations
 - Write a short method

12

Variable Initialization

- Variables must be assigned to before they can be used.

```
public Train (double left, double trackHeight, DrawingCanvas canvas) {
    car = new FilledRect(left, trackHeight - CAR_HEIGHT -
        WHEEL_SIZE, CAR_WIDTH, CAR_HEIGHT, canvas);
    ...
    smokestack = new FilledRect (left + CAR_WIDTH - 2 *
        SMOKESTACK_WIDTH, car.getY() - SMOKESTACK_HEIGHT,
        SMOKESTACK_WIDTH, SMOKESTACK_HEIGHT, canvas);
    smoke = new FilledOval (smokestack.getX() +
        (SMOKESTACK_WIDTH - SMOKE_WIDTH) / 2,
        smokestack.getY() - 1.5 * SMOKE_HEIGHT, SMOKE_WIDTH,
        SMOKE_HEIGHT, canvas);
    ...
}
```

13

NullPointerException

- Occurs if variable is used before it is assigned to.

```
public Train (double left, double trackHeight, DrawingCanvas canvas) {
    smokestack = new FilledRect (left + CAR_WIDTH - 2 *
        SMOKESTACK_WIDTH, car.getY() - SMOKESTACK_HEIGHT,
        SMOKESTACK_WIDTH, SMOKESTACK_HEIGHT, canvas);
    car = new FilledRect(left, trackHeight - CAR_HEIGHT -
        WHEEL_SIZE, CAR_WIDTH, CAR_HEIGHT, canvas);
    ...
}
```

Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
at Train.<init>(Train.java:46)

14

Constructors & Instance Variables

- Instance variable initializations cannot depend on statements in the constructor

```
private smokestackTop = car.getY() - SMOKESTACK_HEIGHT;
public Train (double left, double trackHeight, DrawingCanvas canvas) {
    smokestack = new FilledRect (left + CAR_WIDTH - 2 *
        SMOKESTACK_WIDTH, smokestackTop,
        SMOKESTACK_WIDTH, SMOKESTACK_HEIGHT, canvas);
    car = new FilledRect(left, trackHeight - CAR_HEIGHT -
        WHEEL_SIZE, CAR_WIDTH, CAR_HEIGHT, canvas);
    ...
}
```

Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
at Train.<init>(Train.java:41)

15

Declaring Duplicate Variables

- Be careful not to re-declare a variable

```
private FilledRect car;
public Train (double left, double trackHeight, DrawingCanvas canvas) {
    FilledRect car = new FilledRect(left, trackHeight - CAR_HEIGHT -
        WHEEL_SIZE, CAR_WIDTH, CAR_HEIGHT, canvas);
    ...
}
public boolean contains (Location point) {
    if (car.contains (point)) {
        return true;
    }
}
```

Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
at Train.contains(Train.java:66)

16

Null

- A special value that a variable / parameter can have that means "no value"
- Not applicable to int, double or boolean
- Sending a message to a variable when its value is null results in NullPointerException
- Can check for null:

```
if (var == null) {...
```

17

Checking for null

- Can use an if-statement to determine if a variable has a non-null value

```
if (car == null) {  
    smokestackTop = 100;  
}
```

18

Anatomy of a While Loop

```
while (penguin.isWiderThan (whichHole)) {  
    penguin.turn (LEFT, 1 revolution);  
    whichHole.resize (1.1);  
}
```

Condition

Statements to repeat

- Evaluate condition
- If true, execute statements, evaluate condition, execute statements, evaluate condition, etc.
- If false, skip to statement after while loop

19

Drawing Train Tracks

```
public void begin () {  
    new FilledRect(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT,  
        canvas).setColor(GROUND_COLOR);  
    double tiePosition = 5;  
  
    while (tiePosition < SCREEN_WIDTH) {  
        new FilledRect(tiePosition, TIE_TOP, TIE_WIDTH,  
            TIE_LENGTH, canvas).setColor(TIE_COLOR);  
        tiePosition = tiePosition + TIE_WIDTH + TIE_SPACING;  
    }  
  
    new FilledRect(0, TRACK_TOP, SCREEN_WIDTH, RAIL_WIDTH,  
        canvas).setColor(RAIL_COLOR);  
    new FilledRect(0, TRACK_TOP + GAUGE, SCREEN_WIDTH,  
        RAIL_WIDTH, canvas).setColor(RAIL_COLOR);  
}
```

20

Summary

- Keys to implementing a loop
 - What should the loop do?
 - What should happen each time through the loop?
 - Under what condition should the loop continue?
 - What update happens inside the loop to ensure the loop eventually ends?
 - What initialization must happen before reaching the loop?

21