

COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

Hello!

James (Jim) Young

young@cs.umanitoba.ca

jimyoung.ca

office hours T / Th: 17:00 – 18:00

EITC-E2-582

(or by appointment, arrange by email)

Can we make a command like..

```
moveBadGuy(x, y, xmin, xmax, ymin, ymax)
```

```
moveBadGuy(badGuy1X, badGuy1Y, 0, width-1,  
0, height-1)
```

No – since data is only copied in, any changes that happen in that function are not reflected back in our variables. It is thrown away.

What functions give us data back?
What do they look like?

max, min, random...

```
int result = max(10,4);
```

Functions can only return one piece of data

They can give you an integer,

A float

A string

Etc.

Send data back from a function:

let's make a function `myMax(int a, int b)` which gives us an integer to represent the largest of the two:

```
int bigger = myMax(5, 2); // expect 5 to be the answer
```

first – let's implement this using the tools we already have

we can calculate important information – but how do we send it back?

Send data back from a function:

```
returnType functionName (parameterType parameterName)
```

```
int myMax(int a, int b) {  
    int result = a;  
    if (b>a)  
        result = b;  
    return result;  
}
```

the **return** command does two things:

- it ends the function and returns to where it was called from
- it passes data along from the function to the caller

user-defined functions: syntax

```
int myMax(int a, int b) {  
    int result = a;  
    if (b>a)  
        result = b;  
    return result;  
}
```


these are “local variables”, only exists within the function. this name is not related to how you can use the function

```
...  
int max = myMax(10,20);
```



user-defined functions: syntax

```
int myMax(int a, int b) {  
    int result = a;  
    if (b>a)  
        result = b;  
    return result;  
}  
  
...  
int max = myMax(10,20);
```



Exercise:

Make a function to calculate the distance between two points.

another XKCD comic

remember random()??

how would you implement your own? tough..

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
              // guaranteed to be random.  
}
```

Back to our badguy example

```
final int MAX_MOVE = 20;
final int BG_COLOR = 0;
int badGuy1Size = 20;
int badGuy1Color = 255;
int badGuy1X = 0;
int badGuy1Y = 0;
```

```
int badGuy2Size = 40;
int badGuy2Color = 100;
int badGuy2X = 0;
int badGuy2Y = 0;
```

```
int badGuy3Size = 5;
int badGuy3Color = 180;
int badGuy3X = 0;
int badGuy3Y = 0;
```

```
void setup()
{
  size(500,500);
}
```

```
void drawBadGuy(int x, int y, int size, int col)
{
  fill(col);
  stroke(col);
  rect(x, y, size, size);
}
```

```
void draw()
{
  background(BG_COLOR);
```

```
  // bad guy 1
  int move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
  badGuy1X += move;
  badGuy1X = min(badGuy1X, width-1);
  badGuy1X = max(badGuy1X, 0);
```

```
  move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
  badGuy1Y += move;
  badGuy1Y = min(badGuy1Y, height-1);
  badGuy1Y = max(badGuy1Y, 0);
```

```
  drawBadGuy(badGuy1X, badGuy1Y, badGuy1Size, badGuy1Color);
```

```
  // bad guy 2
  move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
  badGuy2X += move;
  badGuy2X = min(badGuy2X, width-1);
  badGuy2X = max(badGuy2X, 0);
```

```
  move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
  badGuy2Y += move;
  badGuy2Y = min(badGuy2Y, height-1);
  badGuy2Y = max(badGuy2Y, 0);
```

```
  drawBadGuy(badGuy2X, badGuy2Y, badGuy2Size, badGuy2Color);
```

```
  // bad guy 3
  move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
  badGuy3X += move;
  badGuy3X = min(badGuy3X, width-1);
  badGuy3X = max(badGuy3X, 0);
```

```
  move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
  badGuy3Y += move;
  badGuy3Y = min(badGuy3Y, height-1);
  badGuy3Y = max(badGuy3Y, 0);
```

```
  drawBadGuy(badGuy3X, badGuy3Y, badGuy3Size, badGuy3Color);
}
```

How can we simplify this?

```
// bad guy 1
int move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
badGuy1X += move;
badGuy1X = min(badGuy1X, width-1);
badGuy1X = max(badGuy1X, 0);

move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
badGuy1Y += move;
badGuy1Y = min(badGuy1Y, height-1);
badGuy1Y = max(badGuy1Y, 0);
```

See similarities? What if we call a function once for X and once for Y... something like

```
badGuy1X = doMove(badGuy1X, 0, width-1);
badGuy1Y = doMove(badGuy1Y, 0, height-1);
```

Make the function

```
int move = (int)(random(MAX_MOVE*2)-MAX_MOVE);
badGuy1X += move;
badGuy1X = min(badGuy1X, width-1);
badGuy1X = max(badGuy1X, 0);

...
badGuy1X = doMove(badGuy1X, 0, width-1);
badGuy1Y = doMove(badGuy1Y, 0, height-1);
```

Header:

```
int doMove(int position, int minimum, int maximum)
```

Body:

copy the above but use the local variables

Rewrite the main code

Avoid using non-final globals in a function!!

- hard to keep track of
- doesn't scale well to larger programs

A function should accept any changing, specific data as parameters

A function should only return data through the return mechanism.

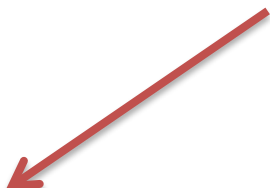
no side effects

Only use global constants

get data back from a function call by “return”ing it!

```
double addTax(double price) {  
    price = price * TAX_RATE;  
    return price;  
}
```

Specific data
comes in!



Preferred way for data to come back!



function parameters and variables...

data is always copied

```
double addTax(double price) {  
    price = price * 1.12;  
    return price;  
}
```

the information is copied
here.

```
void draw()  
{  
    double lunch = 5.00;  
    println(addTax(lunch));  
    println(lunch); // what is output?  
}
```

the information is copied
here. inside the method you
only have a copy. changes here
do not reflect back!

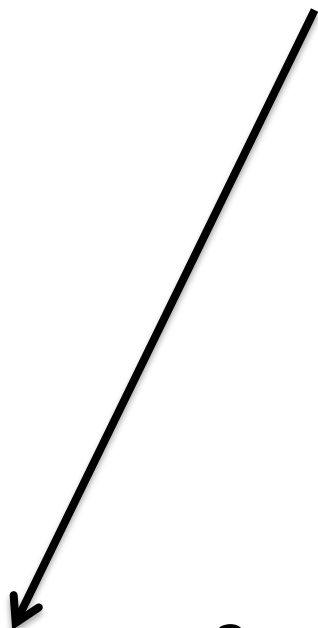
function parameters and variables...

data is always copied

```
double addTax(double price) {  
    price = price * 1.12;  
    return price;  
}
```

```
void draw()  
{  
    double lunch = 5.00;  
    println(addTax(lunch));  
    println(lunch); // what is output?  
}
```

the output is 5.0. Although the function modifies the price variable, since only a copy was passed in, the original was unchanged!



keep tunnel vision.....

forget the rest of your program and solve only the simpler problem in front of you.....

```
double addTax(double price) {  
    //....  
    return?  
}
```

think: I have “price”, and need to return a double
The rest of the program is irrelevant!

functions can call each other!

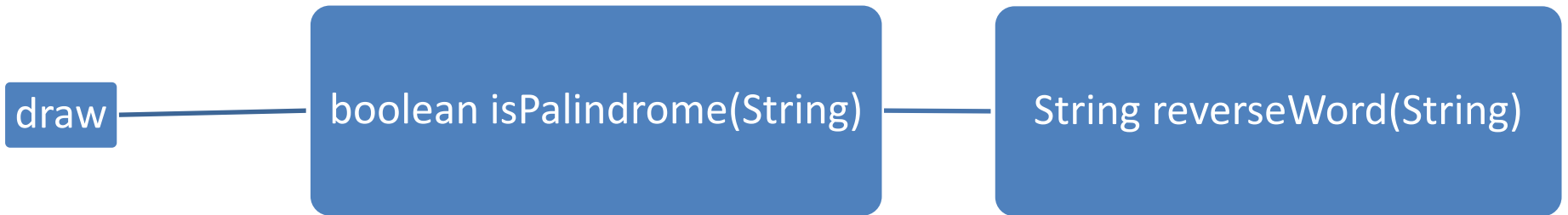
you can call user-defined methods from anywhere – even inside other methods!

make a program that tests if a string is a palindrome!

palindrome: a word that is spelt the same forwards as backwards

algorithm: reverse a word and compare it to the original

program structure



Divide and conquer

divide and conquer

Divide and Conquer: a great way to break up a complex programming problem into smaller steps

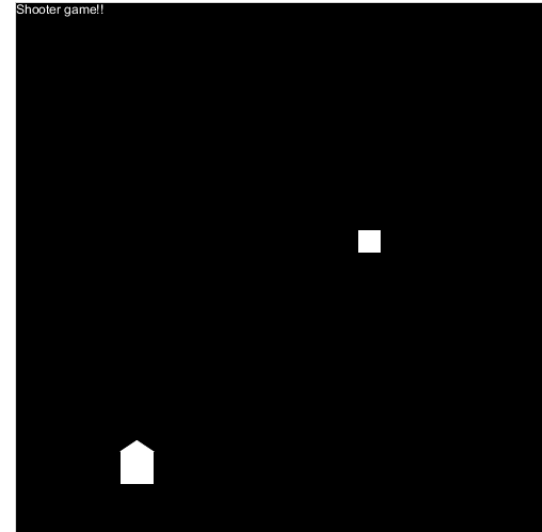
as your programs get longer and longer, this provides a method to enable you to focus on one piece at a time.

Example problem: spaceship shooter

The space ship is linked to the mouseX.
If the mouse button is pressed, the ship
fires a bullet up.

If the user clicks while a bullet is flying,
nothing happens.

If a bullet hits the bad guy, game over



Divide and conquer:

- think about the problem in English first.

- turn these steps into new commands

- focus on one element at a time

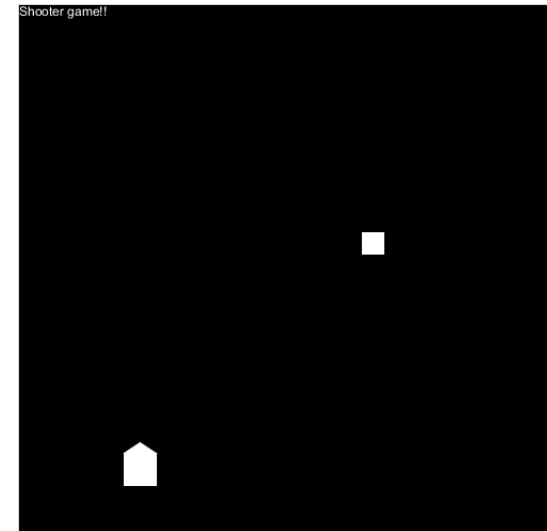
Let's implement our sample problem

step 1: write the program as a series of steps in comments, in English

2: turn each step into a function name (command)

3: create the empty functions

4: start implementing the functions



Steps...

- Draw a title at the top of the screen
- Move and Draw the spaceship
- Check If the spaceship should shoot, and if so, start a bullet
- Move and Draw the Bullet
- Move and Draw the bad guy
- Check if the bullet hit the bad guy, and if so, stop the game
- Draw “WIN” if the game is over.

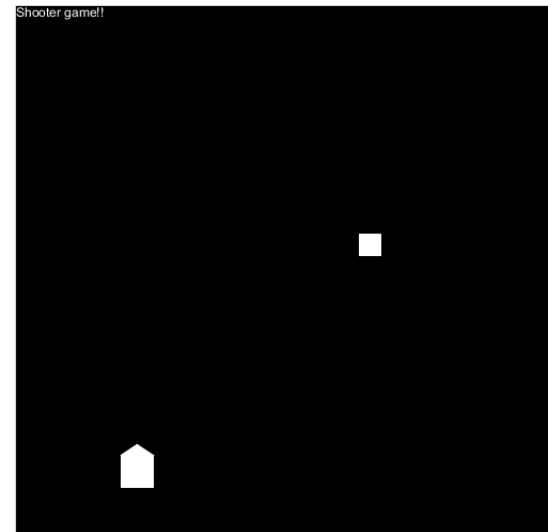
Let's implement our sample problem

step 1: write the program as a series of steps in comments, in English

2: turn each step into a function name (command)

3: create the empty functions

4: start implementing the functions



Turn these steps into commands

```
//Draw a title at the top of the screen  
drawTitle();
```

```
//Move and Draw the spaceship  
moveAndDrawShip();
```

```
//Check If the spaceship should shoot, and if so, start a bullet  
checkAndDoShoot();
```

```
//Move and Draw the Bullet  
moveAndDrawBullet();
```

```
//Move and Draw the bad guy  
moveAndDrawBadGuy();
```

```
//Check if the bullet hit the bad guy, and if so, stop the game  
checkBulletHit();
```

```
//Draw "WIN" if the game is over.  
drawWinMessage();
```

Let's implement our sample problem

step 1: write the program as a series of steps in comments, in English

2: turn each step into a function name (command)

3: create the empty functions

4: start implementing the functions

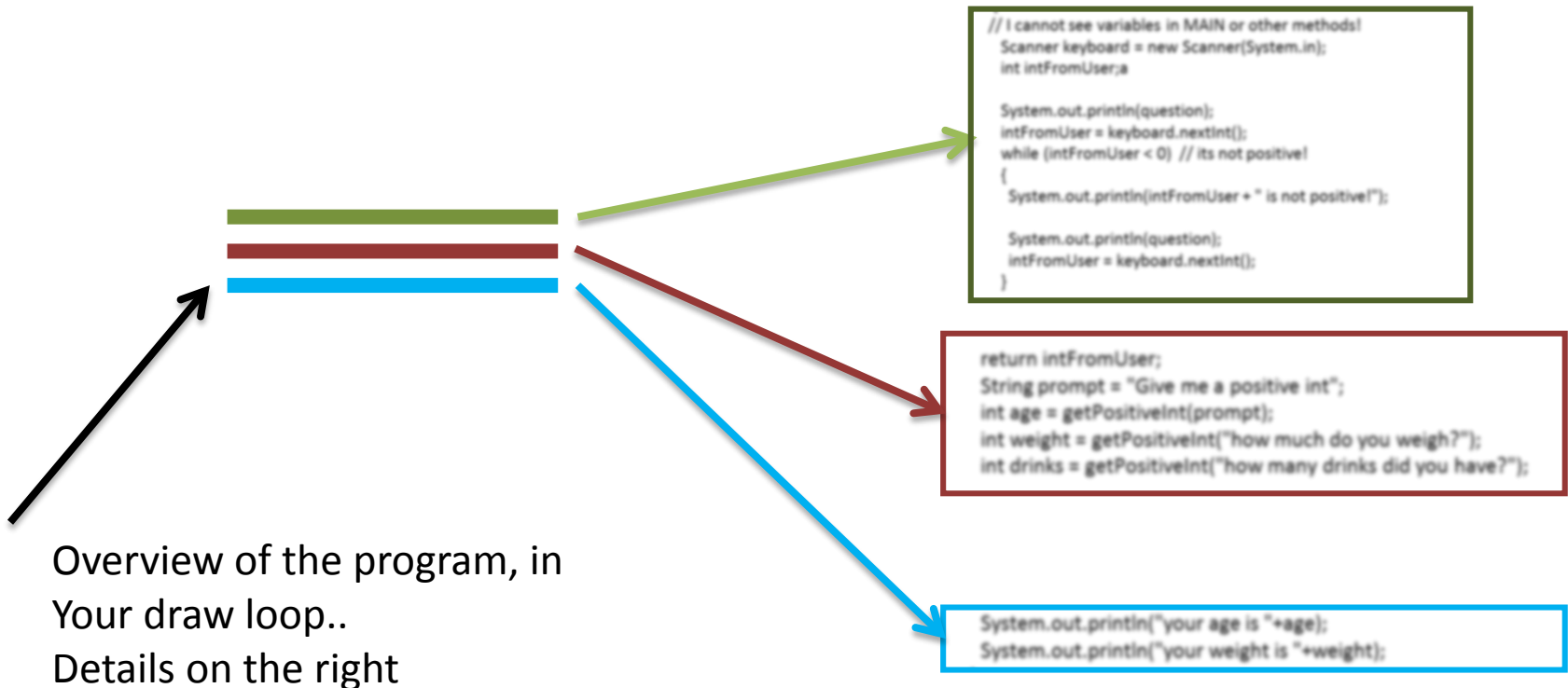


Shooter game!



Assuming these commands work, everything makes sense

Think of the program as a high-level flow



Let's implement our sample problem

step 1: write the program as a series of steps in comments, in English

2: turn each step into a function name (command)

3: create the empty functions

4: start implementing the functions

