Introduction To Programm

check mymodule.tcd.ie Kenneth dawson

www.scss.tcd.ie/course modules/cs1010.
help@scss.tcd.ie
NO LAPTOPS

2nd password for computer Science

coursework worth 20%.

Algorithm - series of steps (sequence) to solve problems
Selection - different actions
itteration - doing something repeatedly

```
public class Hello {
public static void main(String[] args)
    {
        // input code          - information from user
        // computation
        // output code
    }
}
```

Function

- does something for you eg. System.out.print ("What is your name?").
- Sometimes return a value to you.
  eg. String name = input.next();

2.

```
import java.util.Scanner;

public class HelloName {

    Public static void main (String[] args)
    {
        System.out.print ("What is your name?");
        Scanner input = new Scanner (System.in);
        String name = input.next ();
        System.out.println ("Hello" + name);
    }
}
```

Functions

Functions do something

- eg System.out.print ("What is your name?");

Sometimes return a value

- eg String name = input.next();

- Commonly referred to as methods (or member functions)
- Can do very complex tasks
- Allows us to break problems down into subproblems
  - ie put off solving it till later
- Allow us to reuse a general purpose solution...

Input and Output - System.out

- System.out - Stream for output to the console defined in advance by Java
- Also defined are - System.in (for input from the console)
  - System.err (for outputting error message to the console)

② Area vertices of triangle + calculate area

```
import java.lang.Math;
import java.util.Scanner;
import javax.swing.JOptionPane;

public class triangle {

    public static void main(String[] args) {

        String input = JOptionPane.showInputDialog("Enter your three
                coordinate points (all points seperated by a space);
        Scanner = new Scanner(input);
        Double Ax = scanner.nextDouble();
        Double Ay = scanner.nextDouble();
        Double Bx = scanner.nextDouble();
        Double By = scanner.nextDouble();
        Double Cx = scanner.nextDouble();
        Double Cy = scanner.nextDouble();

        Double brackets= ((Ax*(By-Cy) + Bx*(Cy-Ay) + Cx*(Ay-By)) / 2);
        Double abs (Double)
        Maths.abs(brackets)= area

        JOptionPane.showMessageDialog("With your coordinates of :"
        +Ax+Ay+","+Bx+By+","+Cx+Cy+"," + "\n your
        area is :"+area"

    }
}
```

Tutorial Problem (b) average + standard deviation

```
import java.util.Scanner;
import javax.swing.JOptionPane;
import java.lang.Math;

public class averages {

    publin static void main(String[]) {

        String input = JOptionPane.showInputDialog("enter your three numbers:
                ( number1 : number 2 : number 3)");
        Scanner scanner = new Scanner (input);
        scanner.useDelimeter (":");
        double number1 = scanner.nextDouble();
        double number2 = scanner.nextDouble();
        double number3 = scanner.nextDouble();
        double average = (double) [(number1 + number2 + number3)/3];

        double topLine = (number1+number2+number3) - average)
        double bottomLine = (topLine* topLine) / 3)
        double standardDeviation = Math.sqrt (bottomLine);

        JOptionPane.showMessageDialog (null, "with your three numbers:"
            + number1+", "+ number2+", "+ number3+", " "\n your average
            is"+ average + "\n your standard deviation is: "+standardDeviation"
        }
    }
```

Double topLine = $(\text{number1} - \text{average})^2 + (\text{number2} - \text{average})^2 + (\text{number3} - \text{average})^2)/3$

3 26/9/12

Provide various methods (also known as functions)
- print() for printing message
- println() for printing a message and moving to the next line
- printf() for formatting printing

Input and output - Scanner objects
- Used to parse Strings of characters (extracting what you need)
- Defined in a library (java.util.Scanner)
- Can be initialized from
  - System.in - gets it's input from console
  - Strings - parses a string of characters no matter where it's from came
- Provides various methods
  - next() for obtaining the next token - figure/letter separated by space
    - returns a String
  - nextDouble() for obtaining the next double precision number (e.g. 10.293)
  - nextInt() for obtaining the next integer (whole number)

Input and output - JOptionPane
- Used to support simple interact with user. Using "dialog boxes"
- Defined in a library (javax.swing.JOptionPane)
- Provides various methods...
  - showInputDialog() for presenting a simple question and getting a simple typed answer
  - showMessageDialog() for present a message

4

General-simple types

- Need to be able to store data of various types

type	example	operators - add words together.
String	"Hello"	+
double	12.43	+ - * /
(integer) int	345	+ - * / % — gives remainder when number is rounded div

5/2 = 2

5%2 = 1

If we want to store data we can declare variables of any type (or class) and assign values to them.

```
double radius = 14.0;
double diameter = radius*2;
String my_name = "Ken";
```

-

```
import java.util.Scanner;
public class HelloName {
    public static void main(String[] args) {
        String nameInput = JOptionPane.showInputDialog("What is your name?");
        Scanner = new Scanner(nameInput);
        JOptionPane.showMessageDialog(null, "Hello" + name);
        }
}
```

1/10/12

5.

WORKSPACE IN ECLIPSE

- You need to select File_ Switch Workspace_ other...
- Choose browse
- Select U:\ Java Programming
- Click on
- Click OK

VARIABLES

- We often need to store values so that we can use them later.

```
double height = input.nextDouble();
double bmi = weight / (height*height);
System.out.println("Your BMI is " + bmi);
```

- To do this we use variables.
- Variable definition:
  - <type> <name>;
  - <type> <name> = <expression>;
- Variable names are just a group of letters and numbers and (underscores)
  - E.g. bmi, height, radiusInput

TYPES

Type	Minimum	Maximum	Precision	Sample
byte (8 bits)	-128	+127	exact	56
Short (2 bytes)	-32,768	+32,767	Exact	-21947
int (4 bytes)	-2^{31}	$+2^{31}-1$	exact	1008723
long (8 bytes)	-2^{63}	$+2^{63}-1$	exact	+8394024
Float (4 bits)	$-3.40*10^{38}$	$3.40*10^{38}$	lossy	2.34
double (64 bits)	$-1.79*10^{308}$	$1.79*10^{308}$	lossy	2343423
char	0	65535	exact	"A"
boolean	False	True	exact	true

- repn of integers
- characters (unicode in order)

byte = 8 bits

(Jump now to using 32 bit machine)

```
import java.util.Scanner;
import javax.swing.JOptionPane;

public class Wages {

    public static final int CENTS_PER_EURO = 100;
    public static final int MINUTES_PER_HOUR = 60;

    Delimiter - where one thing ends and one thing starts
```

CONSTANTS

- We often need to use values which never change
- These should be declared as constants. Uppercase only - constant value
- Eg `public static final int CENTS_PER_EURO = 100;`
  - final - does not change
- Many libraries provide constants
  - Eg. in the java.lang.Math library:
    - Math.PI
    - Math.E

A very quick MAX_VALUE Tutorial

- The max and min values are defined as constants
  - e.g. Integer.MAX_VALUE
- What value will test be after the following assignment?

```
int test-value = Integer.MAX_VALUE +5;
```

-2147483644

- Number wraparound need to watch out for this

7

3/10/12 Programming

Tutorial - 2 problems

Lab - details available on mymodule after 1pm

Due next tuesday by 23:59

0 marks for late submission

Expressions

Expressions allow us to make computations using variables, constants etc.

eg. double bmi = weight /(height*height);

ASSIGNMENT OPERATOR

Normal assignment operator

- <variable> = <expression>
- <expression> is evaluated and result is stored in variable

int a=0; a=10

a += 10	equivalent to	a = a+10; (operator)
a -= 10	≡	a = a - 10
a *= 10	≡	a = a*10
a /= 10	≡	a = a/10
a %= 10	≡	a = (a % 10

Operator Precedence:

The order in which operators are applied:

Type	Operators	Associativity
Unary postfix	++ --	Right to left
Unary prefix	++ -- + - (type)	RTL
Multiplicative	* / %	Left to Right
Addition	+ -	LTR
Relational	< <= > >=	LTR
Equality	== !=	LTR
Conditional	? :	RTL
Assignment	= += -= *= /= %=	RTL

associativity determines what order operands get evaluated in if they have same precedence

8

a=5 b=7 c=11 ✓

In lab - use workspace U:
- get demonstrator to show you how to debug using breakpoints

d = c%a = 1 ✓
e = c/a = 2 ✓
f = a+b*c+a; = 87
g = a*b+c*a = 90 ✓
h = b%a*c = 22
a += b+c = 23 ✓
b *= c*b = 539 ✓

- Order in which operators are applied can be changed by using brackets.

int averageAge = age1 + age2/2;
brackets: int averageAge = (age1+age2)/2;

- Types can be changed by (casting)

double averageAge = ((double)(age1+age2)) / 2.0;

d = (a+b)/c = 1/2 0.5 ✓
e = c/b*a = 1.113
c = a+b = 4
a%c = b = 0.4
cast1 = 23.00 ✓
cast2 = 3 ✓
cast3 = -256
cast4 = -1
97

8/10/12 (10)

BOOLEANS AND SELECTION

- Booleans represent true/false value
- Selection is a way of choosing different paths/values depending on some (usually boolean) condition

Booleans

boolean running_late = false

- Booleans are always either true or false
- Booleans often computed using comparison operators = return a boolean value
  - < less than
  - <= less than or equal to
  - > greater than
  - >= greater than or equal to
  - != not equal to
  - == is equal to

```
a = 10          e = true        i = true
b = 12          f = false       j = true&false
c = 22          g = false       k = true
d = 10.001      h = false
```

If Statement

```
if (<condition>)
    <statement block>
else
    <statement block>
```

where <condition> evaluates to a boolean
<statement block> can be a single statement or series of statements enclosed in braces

eg
```
if (number%2 == 1)
    print ("number is odd")
else print ("even")
```
need brackets

9.

MATHEMATICAL FUNCTIONS

- Many useful mathematical functions are provided in "java.lang.Math" library

- Absolute value
  <type> abs (<type>); Math.abs(-13) → 13

- Square root
  double sqrt (double); Math.sqrt (16.0) → 4

- Sine or cosine or tan
  double Sin (double); Math.Sin(3.14) → 0.0016

- Inverse Sine/cosine/tan (arcsin/arctan)
  double asin (double); Math.asin (1.0) → 1.5708

1 old penny = 67 new pennies
1 old shilling = 12 old pennies
1 old pound = 20 old shillings
1 new pound = 100 new pennies

take in pound shilling and penny

The ? : operator

- Referred to as a conditional expression
- <condition> ? <true_expression> : <false_expression>

```
String oddOrEven = (number%2==0) ? "even" : "odd "
                                          otherwise
```

- Just a concise way of writing an if-then-else statement for assignment...

```
int peopleCount = 3;

JOptionPane.showMessageDialog(null, "There" + (peopleCount!=1 ? "are" : "is") +
                    peopleCount + (peopleCount!=1 ? "People" : "person"));
```

Comments

- Single line comments:
  - // compute perimeter for a circle
  - can be put on a line after code
- Extended comment

```
/*
~~~
~~~
*/
```

Often used to comment out sections of code temporarily

10/9/25 ② Programming

BOOLEAN - AND, OR AND NOT

Booleans can be manipulated

- NOT (!)

boolean odd = false

boolean not_odd = !odd;

boolean_value	!boolean_value
true	false
false	true

(two things true at same time)

- AND (&&)
- OR (||)

A	B	A && B	A || B
true	true	true	true
~~false~~ true	false	false	true
false	true	false	true
false	false	false	false

boolean is_student = true

boolean in_school = false

boolean in_university = is_student && !in_school.

Precedence	Associativity	
!	right to left	not
&&	Left to right	and
||	left to right	or
=	Right to Left	equal

e = false ✓

f = true ✓

g = true ✓

h = false ✓

i = true ✓

j = true ✓

k = true ✓

15/10/12 Programming LOOPS

Write a single line of code to output whether has passed or failed for example, "You got 40% so you passed your exam" (Given some percentage)

Prefix and Postfix Increment and decrement

post increment variable ++ add 1 to variable after you used it

averageAge = (averageAge * numberOfPeople++ + newAge)/numberOfPeople

Pre increment ++variable

averageAge = (averageAge * numberOfPeople + newAge)/++numberOfPeople;

add a one before you use it

Post decrement variable --

Pre decrement --variable

Quick Tutorial a=7 b=4 c=4

d=6 ✓ p=76 q=32 r=5

e=1 ✓ s=1 ✓

f=12 t=4 u=32 v=15

For Statements:

for (<initialisation>; <continuation condition>; <increment>)
<statement block>

where <initialisation> = one or more statement seperated by comma
<continuation condition> = boolean expression
<increment> = one or more statement seperated by comma

For statement

The simplest (normal) usage looks like this:...

```
For (int count=0; (count<10); count++)
    {
      System.out.println(((count%2)==0)? "," : "") + count );
    }
```

17/10/12. Can be used as part of a statement/expression... variable++.

```
int newStudentNumber = nextFreeStudentNumber++;
//which is equivalent to

int newStudentNumber = nextFreeStudentNumber;
    nextFreeStudentNumber += 1
```

Can also increment before the statement

```
int newStudentNumber = ++lastStudentNumber;
```

* equivalent to (student number +1)

```
lastStudentNumber += 1
int newStudentNumber = lastStudentNumber;
```

Random Numbers

- Often need to simulate random events
- Need a way of generating (pseudo) random numbers
- In Java you can create a random number generator and obtain number within specified range.

→

```
Random generator = new Random();
int randomNumber = generator.nextInt(100);
int anotherRandomNumber = generator.nextInt(50);
```

how many possible values 0-99

range of possibility 0-49

17/10/12 ⑥

Scope

- You can only refer to variables that are in scope
- Variables (etc) are in scope
  - from the point they are defined
  - until the end of the statement/block they are defined in

```
b = 5        r = 1
e = 10
int r=1
for (int c=0; (c<e); c++)      multiply numbers less than e by the value of e
    r=r*b;
```

b^e

```
int f=1                          n = 4   n!
for (int c=0; (c<n); c++)
    f*=c+1                       new f = f*(c+1)
```

	0	1	2	3	4	5
	~~0~~	~~2~~	~~5~~	~~10~~	~~17~~	26
		3	5	7	9	

f = f*(c+1)

f	0	1	2	3	4
c	1	1	2	6	24

22/5/12

Programming

week 5.

week 5: Iteration
6. error handling
7. [illegible]
8. Selection
9+10 functions
11+12 classes

While example:

compute factorial

```
int factorial = 1;
int i = 2
while (i <= number)
{
  factorial = factorial*i;
  i = i+1;
}
```

for version

```
int factorial = 1
for(int i=2; (i<=number); i++)
{
  factorial = factorial *i;
}
```

One line body:

```
int factorial = 1;
int i = 2;
while (i <= number)
  factorial = factorial *i++;
```

While Statement

while (<continuation condition>)
<statement block>

- If the <continuation condition> is true then execute the <statement block>

- Keep doing this until the <continuation condition> is false

22/10/17 ②

a=10 b=20 m=0 c=4 c=10

when q>0
m=m+b
a=20+0
m=20 20
m=20+20 40
m=40+40 ∴ m=80

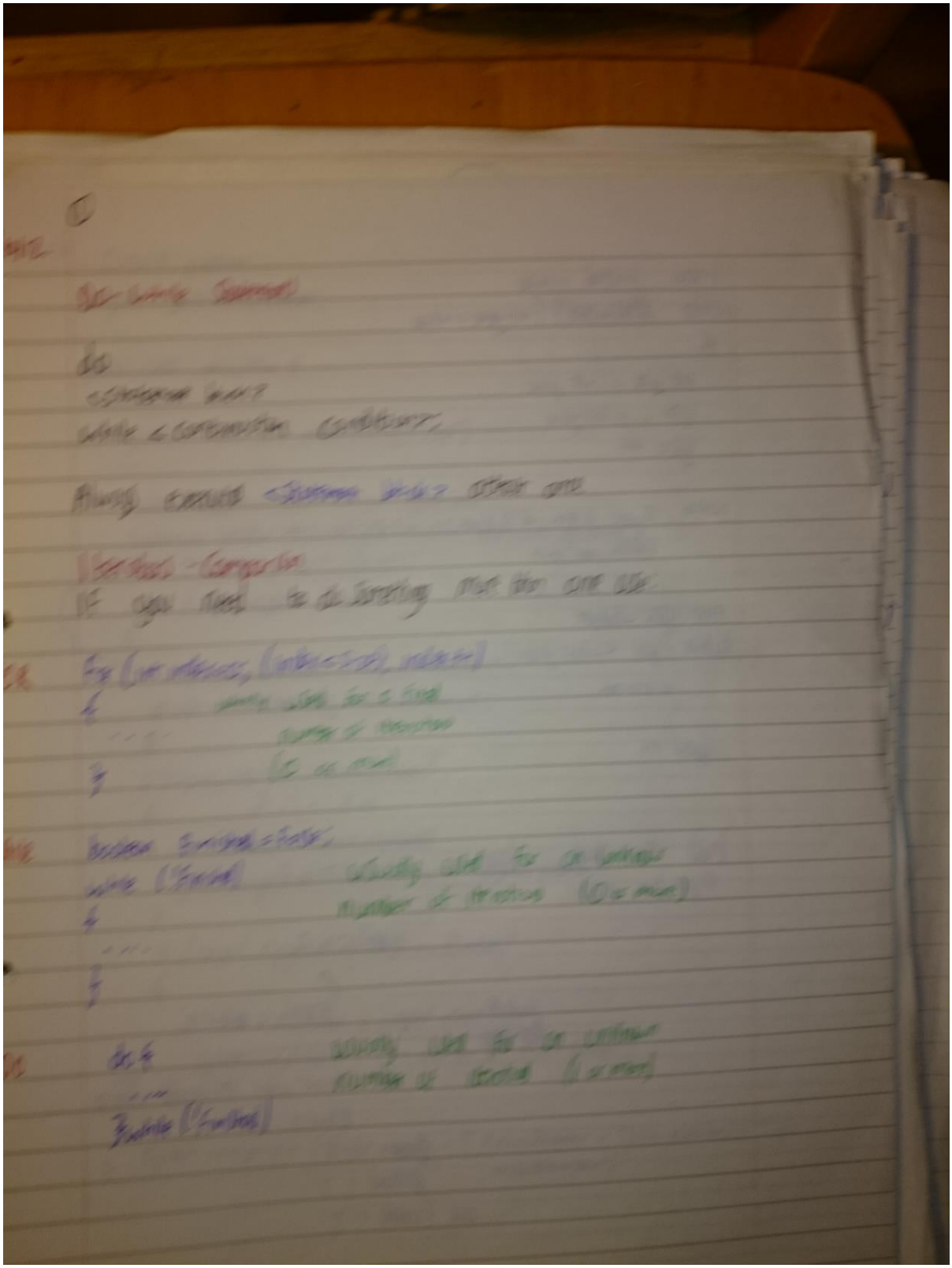
while (c--!=0)
m+=b;

while (c>0){
c=c-1
m=m+b
}

Avoiding errors with Scanner
When Scanner has to input integer but nothing entered

Scanner hasNext" method)
- Scanner provides method which allow us to check whether the input is the right type before we call the "next" method.

- hasNextInt() returns whether there is a next token and whether it is an integer.
- hasNextDouble() return booleans true/false
- hasNext()
- hasNext(<pattern>)
- hasNextByte()



HUGO NILEY

31/10/12 Programming

Week 6 - error handling

- Errorr handlny
- Exception handling

- Handling errod properly makes code makes code much longer (and complex)
- Two main methods:
  - Dealing with errors when they occur.
    - i.e. when input is obtained/ or computations are done
  - Dealing with errors by handling the problems they cause (exception handling)

Exception handling

- Reverse digits program
- What happens if the user clicks 'cancel'?
  - Raises a NullPointerException

- What happens if the user doesn't enter a number and clicks ok?
  - NoSuchElementException raised

Error handling using exception

- We can catch exceptions using a try-catch statement.

```
try
{
  <some code>
}
catch (<exception type> <name>)
{
  <code to run if the exception is caught
}
```

can repeat "catch" block

```
catch
{...
}
```

12/11/12 Week 8 programming Switch Statement.

```
int number;                    break - exits out of loop
Switch (number)
{
 case 1:
  System.out.println("Test 1");
  break;
 Case 2:
 Case 3:
  System.out.println("Test 2 or 3");
  break;
 Case 4:
  System.out.println("Test 4");
 default:
  System.out.println("Test 5");
  break;
}
```

Switch Statements

```
Switch (<expression>)
{case <value>:
    <Statement block usually ending in break;>
 case <value>:
    <Statement " " break;>
...
 default:
    <Statement block " " break;>
}
```

where - <expression> evaluate to some simple value
- <Statement block usually ending in break;> is 0 or more statement often terminated with a break; statement

2.

Break statements.

- An unlabelled break statement terminates the innermost switch, for, while, or do-while statement.
  → where there are nested statements it is current one (the deepest one) which is terminated

- break statements are generally not regarded as good programming other than within the switch statement.

```
switch (month
  case (febuary)
      year %.4 =0? 28:29; break
  case (April, June, "September", November)
      30
      break
  default
      31
```

4/11/12 Programming Week 8

Nested Loops:

- Sometimes we need a loop inside a loop...
- e.g. If drawing an image on screen we need to do the following:

```
For every row of the image
    for every column of the image
        draw the relevant pixel to the screen
```

19/11/12

Function call

eg boolean fiveIsPrime = isPrime(5);

two possibilities

- If the function returns a value.

<variable of return type> = <function name>(<parameters>);

if(isPrime(5)) can put in expression

Or we call always just do...

<functionname>(<parameters>);

- this discard the returned value (if there is one)

isPrime(5)

Functions are identified by name and ~~parameter list~~.

Quick tutorial:

a	✓	✓
b	x	x
c	✓	✓
d	x	x
e	✓	✓
f	~~x~~	x

14/11/12 Week 9 Functions

- Functions → utilities to solve problem
- function calls → throw u use functions
- public static main
- Throwing exceptions
- coding standards 3.0

Introduction to functions

- Functions are utilities which perform defined tasks
  - E.g. println(), nextDouble() showInputDialog().

- We use functions so that
  - code does not have to be duplicated
  - code does not get too long
  - We can easily reuse common utilities
  - we can break problem into subproblems
  - We can test seperate section of code independently

Function Syntax

<return type><function name>(<parameter list>)
{
  <function body>
}

Where

- <return type> is some type (e.g. int) or can be void
- <function name> is part of the function identifier
- <parameter list> is a comma seperated list of <type> <name>
  0 or more parameter declarations seperated by comma.
- <function body> is a serial of statements possibly including 1 or more return statements

19/11/12.

1/12 Programming Week 9

Do all routine start with public static?
- Every program has to start ~~somewhere~~ somewhere.
  public static void main (String args[])
- This is always a public static function called main().
- main() can only involve other public static functions UNLESS it created some objects (which we will look at)

- So for now all your functions are going to have to be declared as public static.
  public static boolean isPrime (int number);
  public static int toPower0f (int base, int exponent);

boolean isIntegerEven (int number)
~~if (int % 2 = 0)~~ 3/2 = 1
~~return~~
~~else~~
~~return -1~~

{
return (number % 2 == 0)
}

boolean bmiCalculator (double weight, double height)
{
return ((weight)/((height)*(height)));
}

21/11/12 ①

Throwing exceptions:

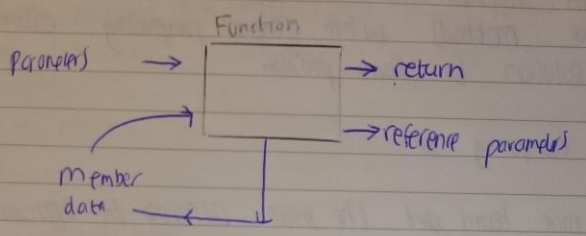
- It is possible to use the exception mechanism to handle 'normal' errors.
- We can throw an exception of a particular type:

```
throw new NumberFormatException();
```

- We can use the existing exception classes
- But we should only do so IF they mean exactly the right thing.
- Otherwise we should create a new exception class.

Programming

26/11/12 Week 10 Functions & Recursion



- Parameter passing - by value
- Iterative approximation.
- Recursion

Parameter Passing

- For simple types the parameters of a function can be used a local variables. Very bad idea

- Any changes made are lost when the function exits

Parameter passing - by reference:

- However when more complex types (objects or arrays etc) are used the changes are retained

- Hence parameter should Never be used as local variables

- The only time that parameter should be changed is if the value of the object / array is supposed to be changed! (more later)

- We need this ability in order to be able to return more than a single value from a function

Iterative Approximation

- iterative methods which generate improving estimates of the solution to some problem

- eg. computing π

- The more terms used the more accurate the approximation

Recursion

- Often an alternative to iteration
- Recursion is the calling of a function within the function itself
- The new function call(s) should represent an easier problem to solve than the original.

- e.g $x^n = (x^{n-1})*(x)$

Week 11 Classes and Objects

- class definition
- Accessing member functions
- Member data
- Constructors
- public vs private
- Static

User defined types:

- ~~Types~~ allow us to store data in different forms

- Classes allow programmers to define their own type
  - These can store data (member data / variables) (data stored belonging to a class)
  - Also provide functionality (methods) (functions), typically relating to the data they store

- Instances of the user defined types (the classes) are called objects

Integer.MAX_VALUE is a class

- Each object has it's own data

Class definition

```
public class <Class_Name> {
    <Add public constants here>
    <add private member data here> (instance data (private))
    < probably add your constructor here >
    <add other methods here>
}
```

Accessing member functions

- Objects of a class are created using "new".

```
ComplexNumber number1 = new ComplexNumber (1.1, 3.1);
```

- Most member functions are accessed by writing:

```
<object_name>.<method_name>(<parameters>);
System.out.println( number1.toString() + "+" + number2);
                    ↑ ↑
                  class object
```

Member data

- Objects store data referred to as member data.
- This data is usually declared at the start of the class definition.
- e.g.

```
private double mRealPart;
private double mImaginaryPart;
```

All member data names should start with a lower case m (for member)

3/12/12 Types + Variables

```
double number2 = 4.4;
double number4 = 8.9;
```

vs classes + objects

invoke instance of complexNumber

```
ComplexNumber number1 = new ComplexNumber (1.1, 3.1);
ComplexNumber number3 = new ComplexNumber (2.4, -1.5)
```

(class) - have method

instance of class) access the data

CONSTRUCTOR

- A constructor is the routine which is used when creating a new object of a class
- It has to initialise all of the member data held by the object
- The constructor is usually passed parameters to tell it what to initialise the member data with...

week 10

28/11/12 Programming

recursion

```
public static int powerOf (int base, int exponent)
{
   int result = 1;
   if (exponent > 0)
      result = base * powerOf (base, exponent - 1);
   return result;
}
```

write Fibonacci Iteratively

3/12/12 Week 11 programming

ACCESSING MEMBER DATA

- To access member data within a method belonging to a class
- For data belonging to the object itself we just use the member data name (i.e. variable name) eg "myImaginaryPart"
- For data belonging to some other object of the same class we use

  <object-name>.<member name>

- For data belonging to the object itself we can also use the keyword **this** (to refer to the current object):

  this. <member name>
  (variable name)

public vs private

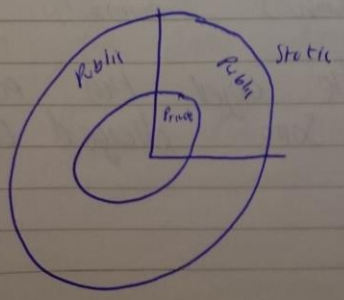
public means that

- It is available to be accessed by code inside or outside the object/class
- If it is a variable it can be changed by any other code
- Use only if you want other code to access it

private means that

- it is hidden from outside the class/object
- Use for internal variables and methods

5/12/12



9/12/12 Programming Week 12.

Can have multiple methods with same name as long as parameters are different.

Public static void Main()

- If a method is "public static" it can be called from anywhere within an object.

- When the first routine in a program is called no other objects have been created so main() is always a public static method.

- Often it is the only method in the class that contains it.

Encapsulation

- Classes are more than just user defined types.

- They can represent complex data (eg a bank record)

- They protect their data from inadvertent misuse.

- Only methods of the class itself can access the private data/method

- Hence the only way other code can alter that data is through the public functions.

- This protection is called data encapsulation.

What does static mean?

```
private static int mHighestAccountNumber = 0;
```

Static means that:

- The variable or method belongs to the class (i.e. is shared by all objects of the class)
- It doesn't need an object to be accessed

  eg. `Account.mHighestAccountNumber`

- If something is **not** declared as static then:
  we can only access it through an object

```
System.out.print(number1.toString() + "+" + number2);
number1.addEqual(number2);
```

Reference

- Objects (which are created using new) are referred to using references

```
ComplexNumber number1 = new ComplexNumber(2.4, -1.0);
```

- We can have many references to the same object:

```
complexNumber number3 = number1;
```

- If we change the object then each reference refers to the same changed object.

Methods creating new objects

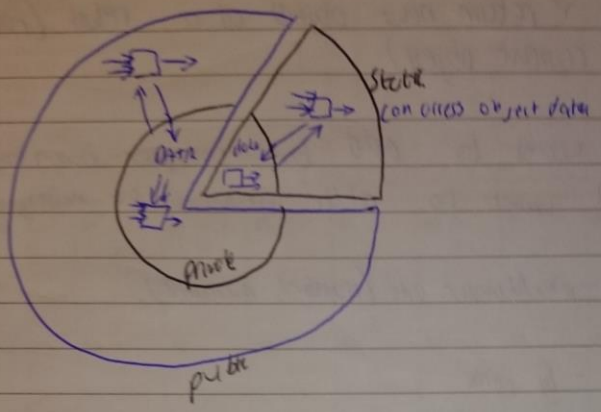
- Methods often return new objects as a result (rather than changing the current object)

- e.g. if we want to add two complex numbers together we may not want to alter either of the existing numbers

```
number 3 = complexNumber.add(number1, number2);
```

Parameter passing - by value

12/12 Programming Week 12.



Copy Constructor:

- It is possible to make a constructor which just copies from an existing object.
- This is called a copy constructor.
- One parameter - an object of the class.
- It must copy values from the passed object to the new (current) object.
- NOTE - member data which refers to an object should never refer to an object which is passed to as a parameter (as this breaks data encapsulation)
  - Instead it must create a copy of the passed object (typically using a copy constructor)

1/11/13. Programming week 3

What are objects?
- Instance of a class
- Data representation (Private)
- Methods

1D Arrays:
Variable size data storage of a particular type

- To refer to a 1D array we need a reference
  `int[] arrayReference = null;` [arrayReference] → null

- We can create an array with values
  `arrayReference = {1,2,3,4};` [arrayReference] → [1][2][3][4]

- Or just say how big it is
  `arrayReference = new int[3];` [arrayReference] → [ ][ ][ ]

- More referred can refer to the same array
  `int[] anotherReference = arrayReference;` [anotherReference]

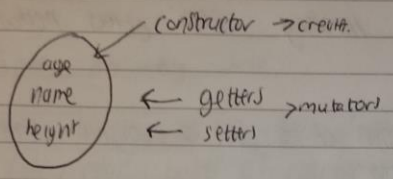
```
public Student record(
  mName = name;
  mIdNumber = idNumber;

  if (passedMarks != null)
  {
    mSubjectMarks = new int[passedMarks.length];
    for (int index=0; index < mSubjectMarks.length; index++)
      mSubjectMarks[index] = passedMarks[index];
  }
  else mSubjectMarks = null;
```

Week 4

4/2/13 Objects and Classes

An object is a component of a program that represents some entity of interest.



Objects of the same type represent the same kind of entity

The attributes of an object describe the entity that it represents

Every object has its own unique name

Object names are usually called object references because they allow objects to refer to other objects

- The value of an attribute can be an object reference
- We can store information about the relationship between objects in the object themselves

Methods

- The methods of an object describe the things that the object can do
  - every object has a set of methods that it is capable of carrying out
  - An object's method describes its behaviour
- Think of methods as commands to which an object responds of responding when ordered

1/2/13 Programming Week 5

Arrays are fixed in Lenght.
ArrayList = Size. ....
Vectas = size = size + 1; size ++

ctrl + space after writing while.

- An array is an indexed collection of values
- Arrays can contain values of any type
- A given array can only contain values of one type

```
public class mainMethod {

    String firstname [4];
    First name = new string{"Vinny", "Paddy", "Siobhan", "Conor"};
    for (int i =0, i <= firstname.length-1, i ++) {
        System.out.println("Friend number" + "i" + "has first name" + firstname[i]);
    }
        string lastname[4];
    String lastname = new String {"cohin", "Nixx", "Clarke", "Jones"};

    for (int z=0, z <=firstname.lenght-1, z ++) {
        System.out.println ("position=" + z + " has the full name" +
                firstname[z] + " " + lastname[z]);
        }
}
```

6/2/13 Programming week 4

Classes:

Method has two part : A heading describe how to use the method.
A body describe how method works

Need to describe - type of result
- calculation to be preformed

Syntax of a java method :

public <result type> <method name> (<parameter>)

Syntax of each java parameter

<type> <parameter name>

Result type is void if no result is produced

Method that initiate new objects are called constructor.
- No return type

2.

- The methods to which an object responds depends on the kind of entity that it represents

Objects interact by invoking each others methods

Classes

- A class is a description of a set of objects that represents the same kind of entity
- Each class describes a particular type of object
- Every object is described by some class
- Every object is an instance of some class
- A class lists the attributes and methods that all objects of the type have in common