RUNSHAW IT & COMPUTING

16th Edition

Java Essentials

Software	Link
JDK	http://www.oracle.com/technetwork/java/javase/downloads/index.html
JDK Install Guide	https://www.youtube.com/watch?v=Hl-zzrqQoSE (The New Boston)
Notepad++	https://notepad-plus-plus.org/
Sublime	http://www.sublimetext.com/
DiaPortable (Class Diagrams)	http://portableapps.com/apps/office/dia_portable

Are you a Future Coder?

Every ____________

__:__

R___

wilkie.n@runshaw.ac.uk

Further Challenges: http://www.practicepython.org/

Maths Challenges: https://projecteuler.net/

Individual Study: https://www.sololearn.com/Course/Java/

Advanced Quizzes: http://quiz.geeksforgeeks.org/

1 Introduction

1.1 Install

Follow instructions from your tutor to install the Java Development Kit. YouTube instructions are available at: https://thenewboston.com/videos.php?cat=31

You will need to:

- Download and install the JDK
- Setup the environment variables

1.2 Blank Program

A Java program must have a class name and a main method. Now is a good time to look at the **Java conventions** on the back of this booklet.

```
public class BlanketyBlank
{ //start of class


    public static void main(String[] args)
    {
        //Add the code here
    }
} //end of class
```

1.3 Compile and Run (& Clear)

```
javac BlanketyBlank.java
```

```
java BlanketyBlank
```

```
CLS
```

2 Basics

2.1 Output

This output statement will display text on the command prompt.

```
System.out.println("Hello World"); //text
```

Note: Println is PrintLN with lowercase letters

```
System.out.print("All ");
System.out.print("on ");
System.out.print("one ");
System.out.print("line");
```

Note: Print does not start on a new line

```
System.out.println("On \n separate \n lines");
```

Note: \n is the return command

```
System.out.println("All \t tabbed \t out");
```

Note: \t is the tab command

```
String name = "James";       //Declare a variable

System.out.println(name);    //Use a variable
```

```
System.out.println("Hello " + name); //concatenate
```

```
System.out.println(4*2);     //calculation
```

HelloWorld.java	
Using only the main method:	
Standard	• Output the following: "Hello World"
	• Output the result of 9/3 using a SOP statement
	• Output the result of 9%3 using a SOP statement
	• Use the concatenation operator (+) to say name each member of your favourite band e.g. "Stone Roses: Ian, John, Reni, Mani"
	• `//Comment your code to get signed off`

AdvancedOutput.java	
Using only the main method:	
	• Create a main method inside your new class
	• Create a String variable called *name* and assign it with the value of your name
	• Write an SOP statement and use the variable to say "Hello *<name>*"
	• Use a hexadecimal value within a SOP and make your program display the denary value `System.out.println(0xF);`
	• Use a binary pattern within a SOP and make your program display the denary value `System.out.println(0b10101010);`
	• `/*Block comment your code` `*to get` `*signed off` `*/`

2.2 Input (String)

```
//Library Class – Import
import java.util.*;
```

```
//Declare a scanner
Scanner inputScanner = new Scanner(System.in);
```

```
//Assign blank variable, then a keyboard value
String name;
name = inputScanner.nextLine();
```

```
//Assign a keyboard value immediately to a variable
String name= inputScanner.nextLine();
```

```
//OPTIONAL
//Make all the characters Upper or Lower case
name = name.toLowerCase();
name = name.toUpperCase();
```

```
//OPTIONAL
//Replace a character within a String
name = name.replaceAll("e","s");
```

```
//OPTIONAL
//Remove any leading or trailing blank space
name = name.trim();
```

```
//OPTIONAL
//select part of String using a start location and
number of characters to get
String partOfName = name.substring(0,2);
```

Conversation.java

Using only the main method:

Standard	• Import the **Scanner** class
	• Declare a **Scanner**
	• Create a String variable called **name** with no value
	• Ask the user for their name as an input and store it as **name**
	• Output a greeting using the **name** variable e.g. "Hello James"
	• Output a follow-up question which mentioned the user's name
	• Give the user a nickname by taking the first 3 letters of their name and adding "gsy" to the end. Output the nickname
	• `//Comment your code to get signed off`

MixedRainbow.java

Using only the main method:

	• Ask the user to enter the colours of the rainbow, one at a time
	• Convert the 1st colour to all lowercase letters, the 2nd colour to all uppercase letters and so on.
	• Replace the letter 'e' in all colours with an 'a'
	• Output the colours
	• `//Comment your code to get signed off`

2.3 If Statement (String)

The Syntax differs for Strings:

Syntax	Description
`name.equalsIgnoreCase("Peter")==true`	Equal to
`name.equals("Peter")== true`	Equal to (Case-Sensitive)
`name.startsWith("P")== true`	First Letter
`name.endsWith("P")== true`	Last Letter
`name.contains("ete")== true`	Within
`name.length()>20== true`	Too Long
`name.isEmpty()== true`	Blank String

```
if (name.equals(differentName)== true)
{
     //string matches this value
}
else if (name.equals("Admin")== false)
{
     //string does not match another variable
}

//Doesn’t need an ‘else’, only 2 options
```

```
if (name.contains("pete")==false)
{
     // join conditions using || for OR
     // join conditions using && for AND
}
else
{
     //name doesn’t have pete in it
}
```

Password.java

Using only the main method:

	• Declare a String variable called **`username`** and set a value
Standard	• Declare a String variable called **`password`** and set a value
	• Ask the user to enter a value for the username and save as **`userEntry`**
	• Ask the user to enter a value for the password and save as **`passEntry`**
	• Compare **`userEntry`** with **`username`**
	• Compare **`passEntry`** with **`password`**
	• Tell the user whether or not they have logged in successfully
	• If wrong, tell the user which part of the log in was incorrect
	• `//Comment your code to get signed off`

ChooseMonth.java

Using only the main method:

	• Ask the user to enter the current month as a number
	• Tell the user the name of the month
	• Tell the user if the month contains the letter “e” or “E”
	• Output whether the current season is Spring, Summer, Autumn or Winter
	• Declare a variable called **days**, to hold a whole number.
	• Within each if statement, set **days** as the number of days in that month, ignoring leap years. Output **days** to the user
	• Ask the user for the current year and use the **MOD** operator to decide if it is a leap year.
	• Test using **`February 2016`** and output the result to the user
	• `//Comment your code to get signed off`

2.4 Data Types & Exceptions

To store data, we declare **variables** *(aka attributes)* with a name and a data type.

The variables can be **declared with blank values:**

```
//blank declarations
boolean paid;       //blank boolean
int age;            //blank int
double cost;        //blank double
String name= "";    //don't have blank strings
char initial;       //blank char
```

or can be given an initial value…

```
//Declarations with value
boolean paid=false;         //Boolean with value
int age = 0;                //Integer with value
double cost = 0.0;          //Double with value
String name = "NotSet";     //String with value
char initial = 'A';         //Character with value
```

To update a variable with **a new value** … (Do not include the data type)

```
//Update a value by hard coding
name = "Elvis"
age = 21;
initial = 'E';
paid = true;
cost = 5000.00;
```

2.4.1 Converting between types

All data should be **collected as a String** and then **converted** to the correct data type, using a **parse** method. Beware, this can throw an **exception.**

```
//Convert FROM String
int  convertToInt        = Integer.parseInt("256");
boolean convertToBool    = Boolean.parseBoolean("true");
double convertToDouble   = Double.parseDouble("23.52");
char initial             = inputScanner.next().charAt(0);
```

```
//Convert TO String
convertedFromInt    = 256 + "";
String convertedFromBoolean = true + "";
String convertedFromDouble  = 23.52 + "";
```

An **exception** is an error which can occur in a program. When converting a String to an integer, a **NumberFormatException** may be thrown.

You need to **'try'** to run your code but **'catch'** any exceptions which occur.

Exception Type	Thrown when…
NumberFormatException	A suitable number cannot be converted from the String

```
try
{
    //ATTEMPTED CONVERSION CODE
    int number          = Integer.parseInt("fred");
    boolean answer      = Boolean.parseBoolean("true");
    double cost         = Double.parseDouble("23.52");
}

catch(Exception exc)
{
    System.out.println("error");
    exc.printStackTrace();
}
```

Album.java

Using only the main method:

Standard

- Create a class called **Album** with the following variables declared inside the main method:

```
title: String
artist: String
shopSection: char
numTracks: int
inStock: boolean
price: double
```

- Ask the user for input and assign a value to each variable
- Output the value of each of the variables on the screen
- `//Comment your code to get signed off`

PersonalData.java

Using only the main method:

- Ask the user for their name and age, capturing both as Strings
- Convert the age to an int **using the parseInt() method**
- Use a Try-Catch block around the parseInt() method to protect the program from a NumberNotFound exception
- Output both items of data correctly
- Force the exception to be thrown by entering `“Fred”` as an age
- `//Comment your code to get signed off`

2.5 Challenge: Dog in the Window

Dog.java	
Using only the main method:	
Standard	Create blank variables with suitable datatypes for the following: • Name • Coat Colour • Eye Colour • LengthCM • WeightKG • pricePennies
	• Ask a user to input values for each of the variables
	• Convert the price into pounds
	• Output all of the data about the dog
	• `//Comment your code to get signed off`
	• Ask a user to enter a command for the dog e.g. "sit"
	• Output a response for the dog **based on the command entered**

2.6 Calculations

Operators can include **add, minus, multiply, divide** and **mod.**

```
int number1=9;          //declare and assign
int number2=2;
```

```
//add //subtract //multiply //divide
int answerPlus = number1 + number2;
int answerMinus = number1 - number2;
int answerTimes = number1 * number2;
double answerDivide = number1 / number2;
```

```
//MOD - leave the remainder
int answerMod = number1 % number2;
```

```
int count=0;
count = count+1;   //or count++;
count = count-1;   //or count--;
```

```
//round a double //100.0 = 2dp, 1000.0 = 3dp
double original=1.2236245543;
double newData=Math.round(original*100.0)/100.0;
System.out.println("To 2 dp: "+newData);
```

```
//Use PI from the Math library
double newData=Math.PI*10.0;
```

SwimmingPool.java	
Using only the main method:	
Standard	• Ask the user for the length of the pool in metres • Ask the user for the width of the pool in metres • Ask the user for the depth of the pool in metres
	• Calculate the perimeter of the pool • Output the perimeter of the pool
	• Calculate the volume of the pool • Output the volume of the pool
	• //Comment your code to get signed off

RoundPool.java	
Using only the main method:	
	The swimming centre has built a new circular pool: • Ask the user for the diameter of the pool
	• Output the circumference of the pool
	• Output the area of the pool
	• MOD the radius by 1, to show the fraction only.
	• //Comment your code to get signed off

2.7 Random Numbers

Random is a class within the **Utilities** package. An import statement is needed at the top of the program in which you wish to utilise it (**above the class name**).

```
//Library Class – Import
import java.util.Random;
```

```
//declare a new obj of Random
Random r = new Random();

//Get a random between 0 and 100
int theRandom= r.nextInt(100);
```

Dividing by an integer to create a double value will cause an incorrect (truncated) answer. Instead, divide by a double value:

```
double theAnswer = 100.00/4.00;
```

RandomDice.java

Using only the main method:

Standard	• Perform a random dice roll to generate a number between 1 and 6
	• Generate 4 random rolls
	• Add the 4 values up and output the average as a decimal

Blackjack, also known as twenty-one, is a comparing card game between a player and dealer, meaning players compete against the dealer but not against other players.

The objective of the game is to beat the dealer in one of the following ways:

- Get 21 points on the player's first two cards (called a "blackjack" or "natural"), without a dealer blackjack;
- Reach a final score higher than the dealer without exceeding 21 **or** let the dealer draw additional cards until their hand exceeds 21.

Blackjack.java

Having completed the standard tasks…

	• Generate 2 cards between 1 and 11 for the user and calculate the total
	• Let the user decide whether to twist or stick
	• Generate 2 cards for the dealer and calculate the total
	• If the dealer is not on at least 17, they must take another card
	• Output the winner
	• `//Comment your code to get signed off`

2.8 IF statement (Non-String)

If the condition is true, the code will be executed. IF statements typically use **operators** within the conditions:

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or Equal to
<=	Less than or Equal to
||	OR
&&	AND

```
if (age>18)
{
     //action
}
else if (age<18)
{
     //action
}
else if (age==18 && birthday==true)
{
     //action
}
```

ExamGrade.java

Using only the main method:

Standard

- Ask the user to enter a test score from 0 - 100
- Calculate their grade based on the following:

A*	**90-100**
A	**80-89**
B	**70-79**
C	**60-69**
D	**50-59**
E	**40-49**
U	**<40**

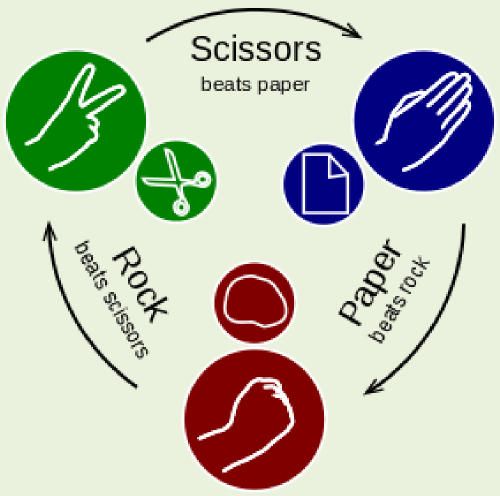
- Output the exam grade
- Validate the program by telling the user when their input is outside of the range 0-100
- `//Comment your code to get signed off`

BiggestNumber.java

Using only the main method:

- Ask the user for 2 integer numbers
- Output which of the numbers is larger
- Instead, ask the user for 3 integer numbers
- Output which of the numbers is the largest
- Using the MOD operator, output whether the largest number is odd or even
- `//Comment your code to get signed off`

2.9 Challenge: Rock, Paper, Scissors



RockPaperScissors.java	
Using only the main method:	
	• Ask the user to choose "rock", "paper" or "scissors"
	• Generate a random computer choice of rock, paper or scissors
	• Compare the values and decide who wins
	• Output the winner
	• `//Comment your code to get signed off`
	• Loop the game until the user types the word 'exit' *(Requires skills from section 3)*
	• Display the number of wins for each player

3 Procedural

3.1 Method Calls

Writing a program using one method is a poor way to program. Instead, we should use the main method **to call other methods.** Methods are chunks of code which each perform a task.

As the main method is *static*, an object must be declared and methods are subsequently called using that object. **Global variables** can also be accessed from any method other than main.

```
public class FirstMethodPrograms
{
     //Global variables here

     //Custom Methods

     //Main Method here
}
```

A method call from Main Method <u>must</u> use the object name:

```
public static void main(String[] args)
{
     Student st = new Student(); //Object declaration
     st.sayName(); //Method calls here
}
```

Object: *An instance of a class*

```
public static void main(String[] args)
{
     Student st = new Student();
     st.startProgram();
}
```

However, a method call from another method must not use the object name:

```
public void startProgram()
{
     sayHello();
     sayName();
     saySubject();
}
```

```
public void sayHello()
{
     System.out.println("Hello!");
}
```

```
public void sayName()
{
     String name="Jane"; //Local variable
     System.out.println("My name is " + name);
}
```

```
public void saySubject()
{
     String subject="Computer Science";
     System.out.println("I study " + subject);
}
```

ModularDog.java	
Using methods where appropriate…	
Standard	• Create a single procedure which will ask for a **name,** store the **name** using a **local variable** and output the **name** back to the user
	Create similar procedures for each of the following: • coatColour • eyeColour • lengthCM • weightKG • pricePennies
	• Create a single procedure to ask for a command for the dog, store the command and output a response for the dog
	• `//Comment your code to get signed off`

ModularDog.java	
Using methods where appropriate…	
	• Alter the program to declare global variables
	• Create a procedure to output the dog's attributes
	• Create a menu system which allows the user to select which attribute of the dog to change
	• `//Comment your code to get signed off`

3.2 Validation

```
//Presence Check
String name = "";


if (name.isEmpty()==true)
{
     System.out.println("Presence Error - Name is
                                              blank");
}

else
{
     System.out.println("Name is OK");
}
```

```
//Length Check
String phone = "01257 542368789";

if (phone.length()>12)
{
     System.out.println("Length Error: Phone # too
                                              long");
}
```

```
//Range Check
int deposit = -1;

if (deposit <0 || deposit >1000)
{
     System.out.println("Range Error: Out of
                                              range");
}
```

```
//Format Check
String email = "studentatcollege.com";

if (email.contains("@")==false)
{
     System.out.println("Format Error: Email has no
                                               @");
}
```

```
//Type Check
String tempAge = "19 and a bit";

try
{
     int age = Integer.parseInt(tempAge);
}

catch (NumberFormatException ex)
{
     System.out.println("Type Exception: Integer not
                                        entered");
}
```

```
//Lookup Check
String year = "Year 11";

if(year.equals("AS") || year.equals("A2"))
{
     System.out.println("OK – Selected from list");
}

else
{
     System.out.println("Lookup Error: Not in
                                            list");
}
```

ModularValidation.java	
Using methods where appropriate…	
Standard	• Create a presence check method
	• Create a length check method with a lower limit
	• Create a range check method with an upper and a lower limit
	• Declare an object and call the 3 methods
	• Alter the methods so they can accept user input
	• Stress test each method and add extra statements as required
	• `//Comment your code to get signed off`

ModularValidation.java	
Continuing with the previous class, using methods where appropriate…	
	• Create a length check method with an upper and a lower limit
	• Create a lookup method
	• Create a type check method for an integer
	• Declare an object and call the 3 methods
	• Alter the methods so they can accept user input
	• Stress test each method and add extra statements as required
	• `//Comment your code to get signed off`

3.3 while Loop

```
//Declare variable to use in the loop
int count = 0;
```

```
while(count<100)
{
	//Code to repeat
	count++;
}
```

```
//Used to pause the loop for x milliseconds
try
{
	Thread.sleep(1000);
}
catch (Exception e)
{
	System.out.println("Sleep Problems");
}
```

```
//Used to terminate a loop immediately
break;
```

ModularSafecracker.java	
Using methods where appropriate…	
Standard	• Create a menu for a text-based game with the following options: 1. Manually set a new 3-digit safe code 2. Randomly generate a 3-digit safe code 3. Set max number of guesses 4. Turn hints on/off 5. Guess the code
	• Option 1: Allow the user to enter a 3-digit code and save this using a suitable data type
	• Option 2: make the computer generate a 3-digit code and save this using a suitable data type
	• Option 5: Create a loop to allow the user to guess the code
	• `//Comment your code to get signed off`

ModularSafecracker.java	
Using methods where appropriate…	
	• Option 3: Allow the user to set a max number of guesses or allow unlimited
	• Validate the user entries to only accept digits
	• Option 4: Allow hints to be turned on. If the user gets a correct digit in the correct place, inform the user
	• `//Comment your code to get signed off`

3.4 for Loop

```
//Declare variable to stop the loop
int max = 100;
String language ="Java";
```

```
//i is the count
//i<max is the condition
//i++ after each loop

for(int i=0; i<max; i++)
{
     //code to repeat
}
```

```
//Used to pause the loop for x milliseconds
try
{
     Thread.sleep(1000);
}
catch (Exception e)
{
     System.out.println("Sleep Problems");
}
```

```
//Used to terminate a loop immediately
break;
```

The methods below are from the Character class and are static. They are called using the name of the class:

Syntax	Description
`if(Character.isDigit('a')==true)`	Number?
`if(Character.isDigit(1)==true)`	Number?
`if(Character.isDigit(tempChar)==true)`	Number?
`if(Character.isLetterOrDigit(tempChar)==true)`	Let/Num?
`if(Character.isLowerCase(tempChar)==true)`	Lowercase?
`if(Character.isUpperCase(tempChar)==true)`	Uppercase?

http://docs.oracle.com/javase/7/docs/api/java/lang/Character.html

```
//This example will print the character at each position
of a String

for(int i=0; i<language.length(); i++)
{
     System.out.println(language.charAt(i));
}
```

ModularNameChecker.java	
Using methods where appropriate…	
Standard	• Create a method to ask the user to enter their name
	• Create a method to output each character of the name on new line using a loop
	• Only output the character if it is a letter, ignore any digits that the user may have entered.
	• Test the program using **Y334O99D56A**
	• `//Comment your code to get signed off`

ModularTenGreenBottles.java	
Using methods where appropriate…	
	• Create a method to countdown the number of bottles on the wall from 10
	• As each bottles falls, tell the user how many are left
	• Extend the program by adding a delay between each number
	• Extend the program to correctly print the song lyrics to the following, using a loop: http://99-bottles-of-beer.net/lyrics.html
	• `//Comment your code to get signed off`

3.5 Challenge: Fizzbuzz

The FizzBuzz challenge is a common programming problem that is often used at interviews.

It uses **repetition and selection** in the same program and relies heavily on the use of the **mod** operator.

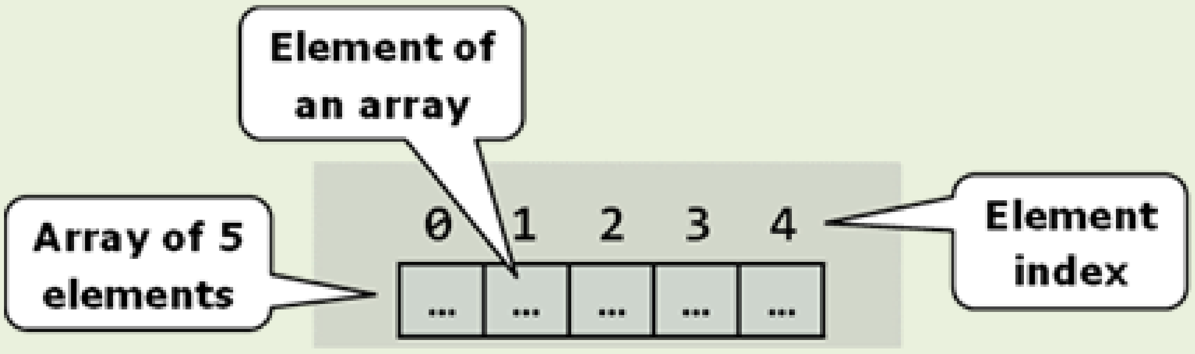
ModularFizzbuzz.java	
Using methods where appropriate…	
Standard	• Create a loop to count from 1 to 100
	Inside the loop, create IF statements which will: • Output **'FizzBuzz'** on numbers divisible by 3 and 5 (mod 15) • Output **'Buzz'** on numbers divisible by 5 • Output **'Fizz'** on numbers divisible by 3 • Output the number for all other numbers
	• `//Comment your code to get signed off`
	• Allow the user to enter the number that they wish to go up to.
	• Count and display the number of Fizz's, Buzz's and FizzBuzz's.
	• Allow the user to enter an alternative number for Fizz
	• Allow the user to enter an alternative number for Buzz
	• Allow the user to enter an alternative number for FizzBuzz
	• `//Comment your code to get signed off`

4 Arrays

4.1 Creating an Array

Arrays store many values as one data structure. An array needs to be declared with a length. Arrays should be declared globally if they need to be accessed in multiple methods.

Access to an array is done via an index e.g. [3]. **Array indexes in Java begin at [0]. An array with a length of 10, would be indexed 0-9.**



```
//A blank array declared with a length of 5
String[] weekDays = new String[5];
```

```
//A blank array - not been given a length
String[] weekDays;
int theSize=5;

//Declaring a length later
weekDays = new String[theSize];
```

```
//Hardcoded array - values are declared
String[] weekendDays = {"Saturday","Sunday"};
```

```
//Assign a value to a position within an array
weekDays[0] = "Monday";
weekDays[1] = "Tuesday";
```

```
//Output a position within an array
System.out.println(weekDays[0]);
```

```
//Optional - Get the length of an array
//NOTE - No brackets on length
int theLength = weekDays.length;
```

```
for(int i=0; i< theLength; i++)
{
     //cycle through an entire array
}
```

```
//OPTIONAL
//Split a string by a character into an array
String primary = "red,blue,yellow";

String[] splitPrimary = primary.split(",");
```

Exception Type	Thrown when…
ArrayIndexOutOfBoundsException	The code tries to use an array index which is invalid.
NullPointerException	The code tries to access an array index, but the location is null.

```
try
{
     //ATTEMPTED CONVERSION CODE HERE
}

catch(Exception exc)
{
     System.out.println("CONVERSION error");
     exc.printStackTrace();
}
```

StarterArray.java	
Using methods where appropriate…	
Standard	• Hard code a global String array containing the days of the week
	• Create a method that loops through the array and prints the contents to the command line
	• `// Comment your code to get signed off`

NumbersArray.java	
Using methods where appropriate…	
	• Create a new global integer array with 5 empty spaces.
	• Ask the user to input a number as a **<u>String</u>**
	• Use **parseInt** to convert the String and save it at the next location in the integer array
	• Add a **Try-Catch** block around the parseInt. • Set a boolean to true value if the try executes successfully. • Set a boolean value to false if it does not • If the boolean value is false at the end, use a while loop to ask for another value
	• Use a loop around the instructions above, to gain 5 valid numbers
	• Output the numbers once the array has been filled
	• `// Comment your code to get signed off`

4.2 Searching an Array

All search algorithms will require an existing array of data

```
//hardcoded array of Strings
String[] ARRAYNAME = {"Cat","Sat","On","Mat"};
```

All search methods will require a value to look for. This can be implemented with a simple global variable or a more difficult technique, using Parameters.

Using a global variable

```
String SEARCHVALUE = "Peter"; //search term
```

```
// Search with a String parameter
public void mySearch()
…
```

```
mySearch(); //no parameter
```

Using a parameter

```
// Search with a String parameter
public void mySearch(String theName)
…
```

```
mySearch ("Peter"); //String parameter
```

4.2.1 Linear Search

```
//Pseudocode for a linear search

FOR (count=0; count< LENGTHOFARRAY; count++)

     IF (ARRAYNAME[i].equals(SEARCHVALUE)) THEN
          OUTPUT ("Match at position " + count);
          break;
     ENDIF

ENDFOR
```

4.2.2 Binary Search

```
//Pseudocode for a binary search


SET start = 0
SET end = LENGTHOFARRAY;
SET found=false;
SET position=-1;
SET midPoint=-1;

WHILE  (found==false)
     SET midPoint = start + ( end - start ) / 2

     IF (ARRAYNAME[midPoint] < SEARCHVALUE) THEN
        SET start = midPoint + 1
     ENDIF

     IF (ARRAYNAME[midPoint] > SEARCHVALUE) THEN
           SET end = midPoint - 1
     ENDIF

     IF (ARRAYNAME[midPoint] == SEARCHVALUE) THEN
           found=true;
           position = midPoint;
           break;
     ENDIF

ENDWHILE
```

LinearNames.java	
Using methods where appropriate…	
Standard	• Declare a String array of size 10
	• Ask the user to input 10 **unique names** and save to the array
	• Ask the user to input a name to search for, perform the search and output the position where it was found
	• Search for a name that doesn't exist and output the result
	• //Comment your code to get signed off

TenTimesBinary.java	
Using methods where appropriate…	
	• Declare a blank int array of size 100
	• Fill each position of the array with a number **10 times** larger than its index e.g. • [0]=0 • [1]=10 • [2]=20 • [3]=30 • [4]=40 • Etc.
	• Create a binary search
	• Search for **350** and output the position
	• Search for **750** and output the position
	• Search for **653** and output the result
	• //Comment your code to get signed off

4.3 Sorting an Array

All sort algorithms will require an existing array of data

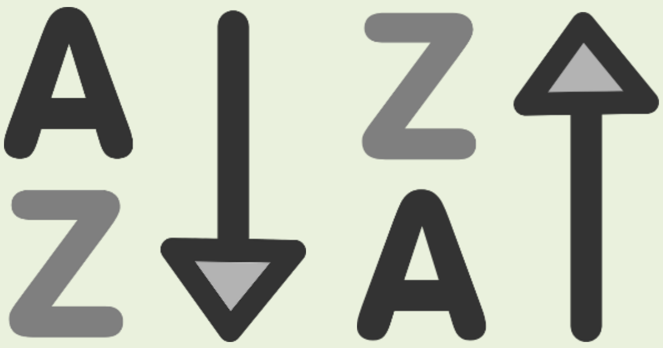
```
//GLOBAL - hardcoded array of integers
int[] ARRAYNAME = {3,5,4,7,1,2,6};
```

A useful way to check the current contents of an array (maybe after each pass) is:

```
//OPTIONAL
//Simplistic way to output the whole array
//Can be used to check after sorting

String textArray=Arrays.toString(ARRAYNAME);

System.out.println(textArray);
```



4.3.1 Bubble Sort

```
//OUTER LOOP through each element ONCE
for(int i=0; i< ARRAYNAME.length; i++)
{
     //INNER LOOP (see below)
}


FOR (outer=0; outer < LENGTHOFARRAY; outer ++)

     FOR (inner=0; inner<LENGTHOFARRAY-1; inner ++)

          IF (ARRAYNAME[inner]>ARRAYNAME[inner+1]) THEN

               String temp = ARRAYNAME[inner];
               ARRAYNAME [inner]= ARRAYNAME [inner+1];
               ARRAYNAME [inner+1] = temp;

          ENDIF

     ENDFOR

ENDFOR
```

4.3.2 Insertion Sort

```
//OUTER LOOP through each element ONCE
//start 1 above index 0
FOR (outer=1; outer<LENGTHOFARRAY; outer++)
     SET currentItem = numbersArray[outer];
     //ascending as outer loop progress

     SET inner = outer-1; //descending from currentItem


     //if item is larger
     WHILE (inner>=0 AND numbersArray[inner]>currentItem)
          //move previously sorted element up
          numbersArray[inner+1] = numbersArray[inner];

          //descend pointer
          inner = inner-1;
     ENDWHILE


     //insert current item in correct place
     numbersArray[inner+1] = currentItem;
ENDFOR
```

BubbleLottery.java

Using methods where appropriate…	
Standard	• Create a lottery ticket for a player by generating six random lottery numbers from **1 - 49**
	• **Before sorting,** output the highest player number
	• **Before sorting,** output the lowest player number
	• Sort the user's numbers in **descending** order **<u>using Bubble Sort</u>**
	• Output all of the numbers for the player's ticker
	• `//Comment your code to get signed off`

InsertionLottery.java

Having completed the standard tasks…	
	• Create a lottery ticket for a player by generating six random lottery numbers from **1 - 49**
	• Make sure that the user has not been given the same number twice
	• Sort the user's numbers in **ascending** order **<u>using Insertion Sort</u>**
	• Output the user's lottery numbers in order
	• Perform a lottery draw by generating six random lottery numbers from **1 – 49**
	• Tell the user how many balls they have matched
	• `//Comment your code to get signed off`

4.4 2D and 3D

Arrays with multiple dimensions are accessed using multiple indexes
e.g. [3][1] (Row 4, Column 2)

Array indexes in Java begin at [0]

```
//Declare an 8x8 2D array
String[][] chessBoard = new String[8][8];

// Declare a 3x3x3 3D array
String[][][] cube = new String[3][3][3];
```

```
//Change values in a 2D array
chessBoard[0][0] = "W Castle";   //row 0 col 0
chessBoard[0][1]= " W Knight";   //row 0 col 1
chessBoard[1][0] = "W Pawn";     //row 1 col 0
chessBoard[1][1]= " W Pawn";     //row 1 col 1
```

```
//Output a single value
System.out.println(chessBoard[0][1]);
```

```
//NOTE – PRINT not PRINTLN
//Output the whole array using nested loops
for (int i=0; i<8; i++)
{
     for (int k=0; k<8; k++)
     {
          System.out.print(chessBoard[i][k]);
     }
     System.out.println();
}
```

```
//Optional – Get the length of an array
//NOTE – No brackets on length
int rows =    chessBoard.length;
int columns = chessBoard[0].length;
```

Minesweeper.java

Using methods where appropriate…

Standard	• Declare a global variable for the **row** and another for the **column**
	• To place the mine, give **row** and **column** a value from **0 – 4**
	• Declare an integer 2D array of size 5 by 5, with **initial values of 0**
	• Display the grid of 0's
	• Ask the user which row and which column they think the mine is in
	• Check if the user guessed correctly and output an appropriate message
	• Update the guessed location with a **'1'** and refresh the displayed grid
	• `//Comment your code to get signed off`

Minesweeper.java

Using methods where appropriate…

	• Allow the user 5 guesses, updating the display after each guess
	After each guess, tell the user if they are: • **Boiling Hot –** The row or column is correct • **Hot –** The row or column is 1 away from the actual • **Warm –** The row or column is 2 away from the actual • **Cold –** The row or column is 3 away from the actual
	• Update the guessed location with a **'1'** and refresh the displayed grid
	• If the user guesses correctly, put an X on the display where the mine was located. Congratulate the user.
	• `//Comment your code to get signed off`

4.5 Challenge: Xando

	Xando.java
Using methods where appropriate…	
Standard	• Declare a String 2D array of size 3 by 3
	• Display the empty board
	• Ask the user to input a name for player 1 and a name for player 2
	• Indicate whose turn it is and allow the user to choose what row and column they want to change.
	• Do not allow a player to choose a position which has already been taken
	• Check the board for a winning combination after each move
	• Tell the user when someone has won.
	• `//Comment your code to get signed off`
	• Allow the users to play another game, tracking how many games each competitor has won
	• Alternate which player starts first
	• Extend by making the game single player vs CPU …*(the level of intelligence is up to you!)*
	• `//Comment your code to get signed off`

5 File Handling

5.1 File Writer

```
import java.io.*; //Library Class – Import
```

```
//Global Variables
String filename = "output.txt";
String fname = "John";
String sname = "Smith";
```

```
//Declared in the method where filewriting occurs
FileWriter fw = new FileWriter(filename);
```

```
//Optional – appends instead of overwriting
FileWriter fw = new FileWriter(filename,true);
```

```
fw.write("This has written to file"); //write text
fw.write("\r\n");  //Move to a new line
```

```
fw.write(fname +","+ sname); //write variables
fw.write("\r\n");  //Move to a new line
```

```
fw.close();
//Called ONCE, when all WRITING IS FINSIHED
```

```
try
{
   //File writing lines go here

   //CLOSE FILEWRITER HERE (after any loops)
}
catch(Exception exc)
{
   System.out.println("WRITE error");
   exc.printStackTrace();
}
```

TextToFile.java	
Using methods where appropriate…	
Standard	• Create a main method with an instance (object) of this class
	• Create a method to write 1 line to a file, using an object of FileWriter
	• Add try-catch blocks within the method as required for FileWriter
	• Call this method from main
	• Adapt the method to write 2 lines to a file on separate lines.
	• `//Comment your code to get signed off`

TextToFile.java	
Using methods where appropriate…	
	• Create a method to ask the user for the name of the file that they wish to write to • Add **`.html`** to save as a web page
	• Find some basic html code from the Internet
	• Write the html code to the file.
	• Ask the user for confirmation that they want to write to file. Only write to file if 'YES'
	• Test the created webpage by trying to open it
	• `//Comment your code to get signed off`

5.2 File Reader

```
//Library Class - Import
import java.io.*;
```

```
String filename = "output.txt";
```

```
BufferedReader br = new BufferedReader
                    (new FileReader(filename));
```

```
// read and store the first line
String tempString = br.readLine();

tempString = br.readLine(); // read the next line
```

```
// loop
     tempString = br.readLine(); // read next line
// end loop
```

```
br.close();
//Called ONCE, when all WRITING IS FINSIHED
```

```
try
{
   //File reading lines go here
   //CLOSE BUFFER HERE (after any loops)
}

catch(Exception exc)
{
   System.out.println("READ error");
   exc.printStackTrace();
}
```

```
//OPTIONAL
Split the string by comma, store in an array
String[] splitData = tempString.split(",");
```

TextFromFile.java	
Using methods where appropriate…	
Standard	• Create a text file with the following data: `James,Jones,01/01/99`
	• Create a method to read a single line from a text file and output it to the user
	• `//Comment your code to get signed off`

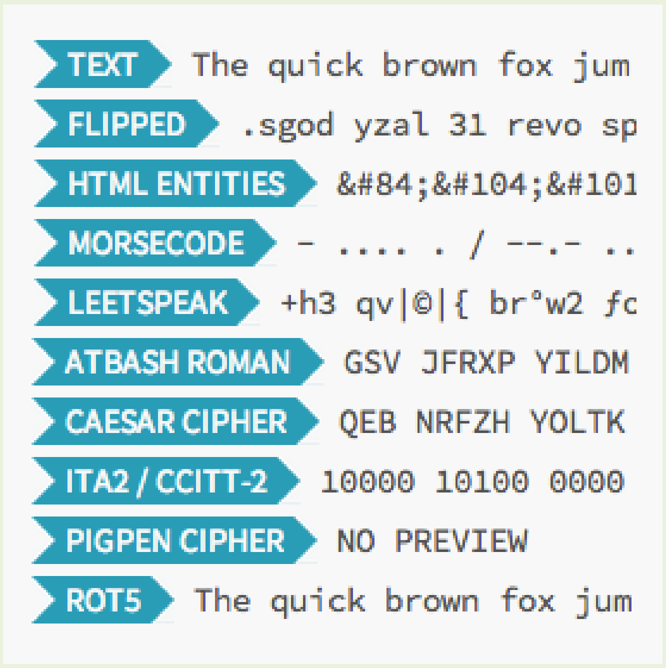
TextFromFile.java	
Continuing with the previous class, using methods where appropriate…	

	• Create a text file with the following data:
	`James,Jones,01/01/99`
	`Sarah,Smith,12/12/99`
	`Bobby,Ball,04/04/99`
	`Harry,Hall,06/06/99`

	• Create a method to ask the user for the name of the file that they wish to read from
	• Use a loop to read and output each line
	• Continue reading while the data read from the file is not null
	• `//Comment your code to get signed off`

5.3 Encryption

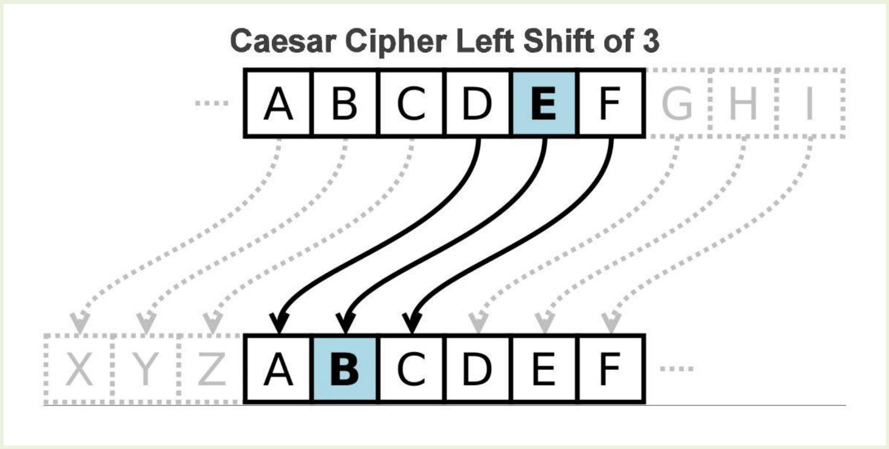
Encryption is the process of altering the content of a message in such a way that only authorised parties can read it. The process requires the original **PlainText,** and the **Key,** which will output the **CipherText.**



TEXT	The quick brown fox jum
FLIPPED	.sgod yzal 31 revo sp
HTML ENTITIES	&#84;&#104;&#101
MORSECODE	- .... . / --.- ..
LEETSPEAK	+h3 qv|©|{ br°w2 fc
ATBASH ROMAN	GSV JFRXP YILDM
CAESAR CIPHER	QEB NRFZH YOLTK
ITA2 / CCITT-2	10000 10100 0000
PIGPEN CIPHER	NO PREVIEW
ROT5	The quick brown fox jum

https://v2.cryptii.com/text/select

One technique for encrypting data is to use a simple **Caeser-Shift** whereby a number is shifted down or up the alphabet. For example, using a shift of **-3**, the letter **E would become B**



Programming an encryption algorithm requires confidence in manipulating a character. Remember, a String is just an array of individual characters.

A character also has an ASCII value, of type integer. You can change a character by adding or subtracting to the ASCII value.

Always use System.out.println to check the values you have retrieved!

```
String myString="Java";

//Store the character at each position of a String
char firstLetter = myString.charAt(0);
char secondLetter = myString.charAt(1);
char thirdLetter = myString.charAt(2);
char fourthLetter = myString.charAt(3);

//Or using a for loop…
char aLetter = myString.charAt(i);
```

```
//Get the ASCII value of a character
char firstLetter = myString.charAt(0);

int firstLetterASCII = (int)firstLetter;
```

```
//Alter the position of an ASCII value
int newLetterASCII = (firstLetterASCII-3);
```

```
//Convert an ASCII value to a character
char newLetter = (char) newLetterASCII;
```

```
String myString="Jav";

//Adding a character to a String
char newLetter='a';

myString=myString+newLetter;
```

SecretCaeser.java	
Using methods where appropriate…	
Standard	• Declare a variable to store "Hello World" as a variable called **`plainText`**
	• Using a loop, cycle through each character in **`plainText`** and output the ASCII value for each character
	• Create a new variable with no value called **`encrypted`**
	• For each character in **`plainText`**, add `10` to the ASCII code and add the character to **`encrypted`**
	• Output the final value of **`encrypted`**
	• `//Comment your code to get signed off`

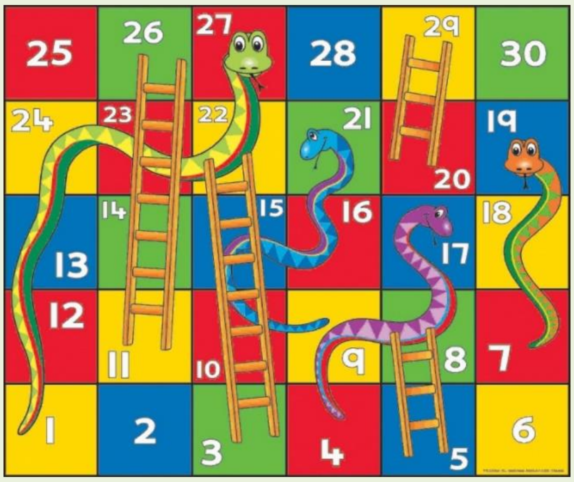
SecretAtbash.java	
Using methods where appropriate…	
	• Ask the user for a sentence
	• Validate the input so it cannot contain a comma
	• Encrypt the sentence using the **Atbash** technique
	• Output the final value of the encrypted data
	• `//Comment your code to get signed off`

5.4 Challenge: Cash Machine

CashMachine.java	
Using methods where appropriate…	
Standard	• Setup a new user with an account number of **12345678**, pin number of **1234** and assign a starting balance of **£1000**
	• Loop a text-based menu with the following options: 1 Login 2 Logout 3 Display Account Balance 4 Withdraw Funds 5 Deposit Funds 6 Exit
	• **Option 1:** The user must enter their account number and pin to be able to use options 3,4 and 5.
	• **Option 2:** Allow a user to logout
	• **Option 3:** Display the balance
	• `//Comment your code to get signed off`
	• **Options 4 and 5:** Automatically write the balance to file after each transaction
	• **Options 4 and 5:** Validate the user entries so the user cannot go overdrawn
	• Encrypt the data when written to file and decrypt it when read back in.
	• `//Comment your code to get signed off`

6 Final Challenges

6.1 Snakes and Ladders



SnakesAndLadders.java	
Using methods where appropriate…	
Standard	• Program a random dice from 1- 6
	• Allow the user to roll the dice and move
	• Create an 1D or 2D array to hold the numbers above
	• Within the array, implement a ladder on the numbers above
	• Within the array, implement a snake on the numbers above
	• Tell the user that they have won when they go past 25.
	• //Comment your code to get signed off

SnakesAndLadders.java	
Having completed the standard tasks…	
	• Extend by making the game multi-player for up to 4 players
	• Allow an exact finish on 25 only
	• //Comment your code to get signed off

6.2 Bingo

Bingo.java	
Using methods where appropriate…	
Standard	Generate a random bingo card for the user: • 12 unique numbers, in ascending order, displayed as 3 rows and 4 columns
	Perform a bingo draw: • Draw unique numbers ranging from 1-90
	Perform a bingo draw: • Output each number when drawn
	• //Comment your code to get signed off

Bingo.java	
Having completed the standard tasks…	
	• Automatically check each number for the user.
	• The program should shout 'Bingo' when the user has collected all of their numbers (full house)
	• Extend by making the game multi player. Indicate which player wins.
	• //Comment your code to get signed off

7 OOP: External Classes

Array of Records (Database)

A database contains a number of records. In Java, a record is a data structure which is capable of holding many attributes with different data types.

To hold many records, another class with an array is also needed.

The following classes would be saved as **BankClient.java** and **CompleteClientList.java**

Whilst **CompleteClientList.java** will compile and run, nothing will happen. Methods still need to be built to allow us to create an instance of **BankClient** and add it to the array

Other methods may be useful such as writing all records to a file, reading all records from a file, searching all records.

```
public class BankClient
{
	String accountNumber;
	String sortCode;
	double balance;
	boolean active;
}
```

A record implemented as a single class

```
public class CompleteClientList
{
	BankClient[] allClients = new BankClient[100];
	int nextClientPoisition=0;

	public static void main(String[] args)
	{
		CompleteClientList cc = new CompleteClientList();
	}
}
```

A list class containing an array

Using External Classes & Calling Methods

Ideally, a central class should be used to call methods from another class. An easy way to achieve this is by the following stages

1. Create the central (interface?) class
2. Within the central class, declare an object of every class you will need to use. Do this globally
3. Use the objects inside to call methods from other classes

Stage	Description

1	`public class TheCentralGUI`
	`{`
	`    //...the rest of the main program`
	`}`

2	`public class TheCentralGUI`
	`{`
	`    ClientList clist = new ClientList();`
	`    BankClient bc = new BankClient();`
	
	`    StaffList slist = new StaffList();`
	`    BankStaff bs = new BankStaff();`
	
	`    //... the rest of the main program`
	`}`

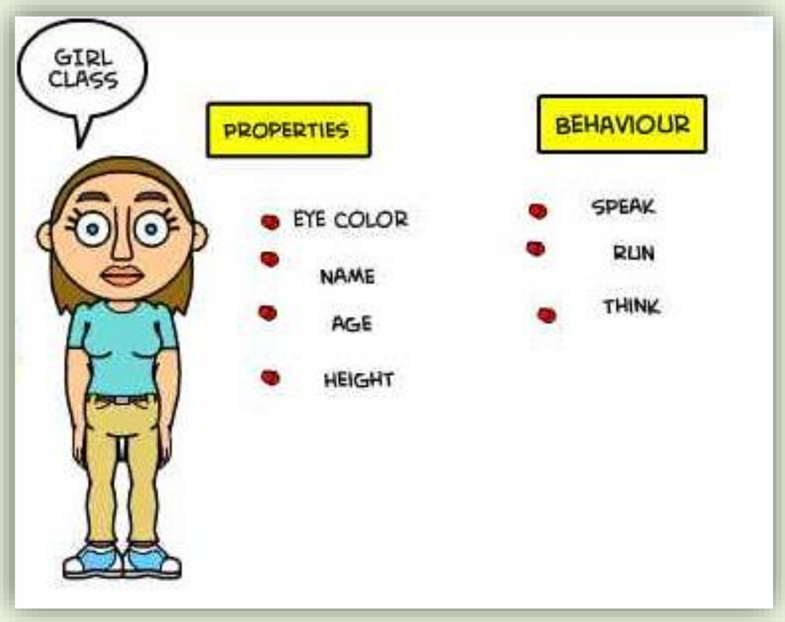
3	`// within the main program`
	
	`    bs = new BankStaff();`
	
	`    bs.name = tfName.getText();`
	
	`    slist.addStaff(bs);`

OOP: The Theory

OOP programs are organized around objects, which contain both **data** and **methods** that can manipulate that data.

Class

A *class* serves as a plan, or blueprint. It specifies the attributes and methods that will be included in objects of that class.



Object

An object an "instance" of a class.

Attributes

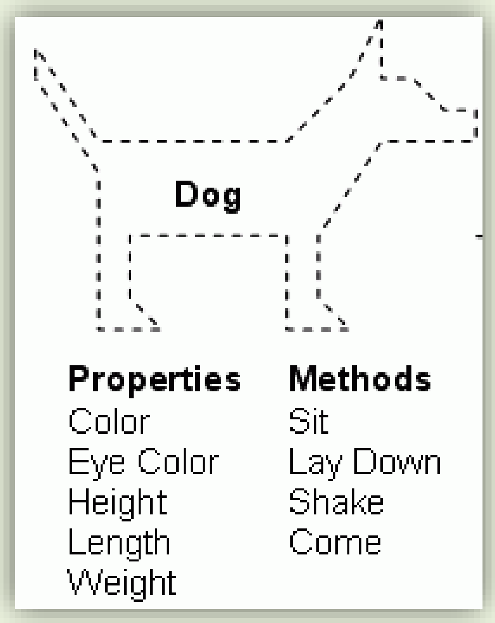
Data to be stored about an object should be declared at the top of the class. These are known as **global variables** and are accessible to the whole class.

- **Local variables** can be declared within a method but are only accessible to that method.

Abstraction

During the process of abstraction, a programmer hides all but the relevant data in order to reduce complexity and increase efficiency, making the problem easier to understand.

This will reduce it to a set of essential characteristics.



Inheritance

Allows a class to be derived from an existing class without modifying it. The derived class has all the data and functions of the parent class, but adds new ones of its own.

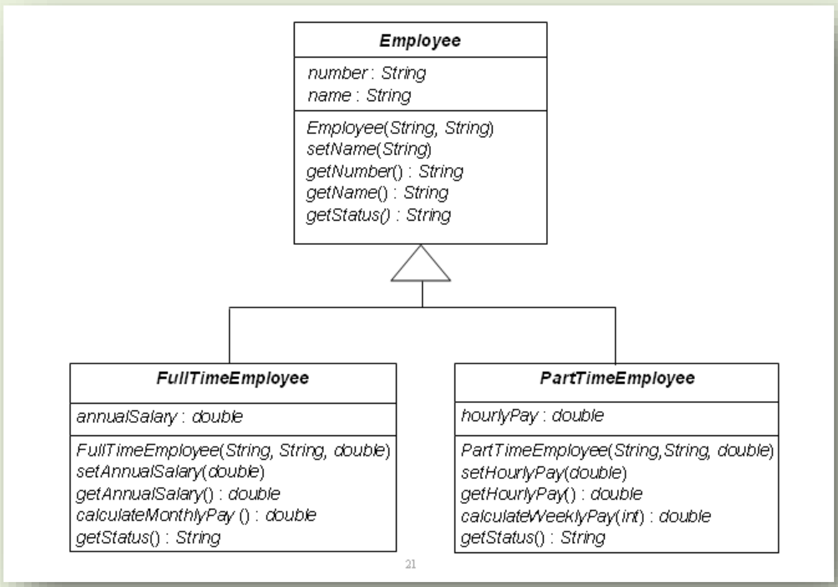


Figure 0-1 - FullTimeEmployee and PartTimeEmployee extend Employee

Some classes are designed specifically to be inherited by other sub classes and objects of such classes cannot be directly instantiated. These classes are called **Abstract Classes.**

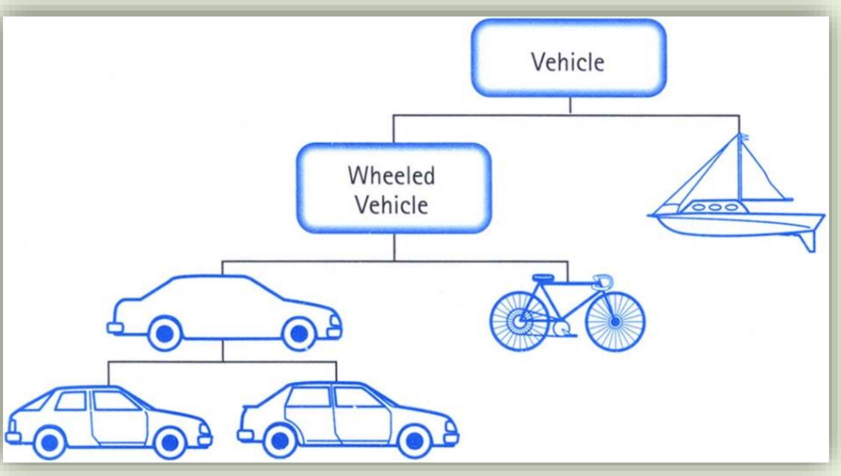
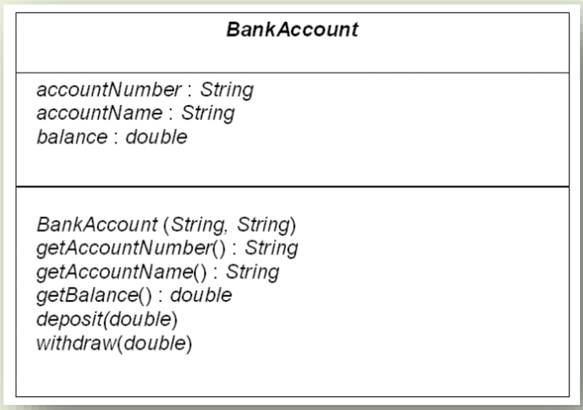


Figure 0-2 - Vehicle and WheeledVehicle are abstract

Encapsulation

The inclusion, within an object, of all the attributes and methods needed for the object.



Access Modifiers

Access modifiers should be used when declaring attributes or methods, using the keyword **private, protected** or **public**.

Private(-) means the attribute will <u>not be accessible</u> by any external classes

- Used for variables in collaboration with Get/Set methods

Public(+) means the method will be accessible by all external classes

Protected(#) means the attribute/method will be accessible by classes within the package

Get/Set Methods

A **'set'** method needs to be public and can be used to change the value of an attribute in this class from another class

A **'get'** methods needs to be public and can be used to retrieve the value of an attribute in this class from another class

Method Types

Procedure

A procedure is a method which does not return a value. A procedure in Java will contain the keyword **void.**

```
// Procedure
public void sayHelloWorld()
{
     System.out.println("Hello World");
}
```

```
// Procedure
public void saySomething(String theMessage)
{
     System.out.println(theMessage);
}
```

Functions

A function is a method which does return a value. A function in Java will replace the keyword **void** with the datatype of the returned data. It will also contain the word **return** at the end of the method.

```
// Function - returns a String
public String checkCompSci()
{
     String answer="Not set";

     if(studyingCompSci==true) //Global boolean?
     {
          answer="Unlucky";
     }
     else
     {

     }

     return answer; //returning as String

```

```
// calling a function from main
// receiving a String

public static void main(String[] args)
{
     Student st = new Student();

     String answer = cst.checkCompSci();
     System.out.println("Your subjects?..." + answer);

}
```

Constructor

A method which is automatically called when an object is created. It is useful to tell the programmer when a new object has been made and sometimes to set the initial value for variables.

It must follow each of the following rules:

- It has **no return type** (~~void~~)
- It begins with a capital letter
- It has with the same name as the class

```
//This constructor will output a statement and assign
a value to the two global variables

public Student()
{
     System.out.println("New Student Created");
     name="Not Set";
     studying="Computer Science";
}
```

```
public static void main(String[] args)
{
     Student st = new Student();
}
```

It can also be useful to call a method automatically:

```
//This constructor will read the file when a list is
made

public StudentList()
{
     System.out.println("New StudentList Created");
     readStudentFile();
}
```

Other Topics

Topic	Description
ArrayList	Similar to arrays, but contain pre-built methods for manipulating data in the list
StringBuilder	Similar to the String datatype, but uses objects which are **mutable.** Useful methods include append( ) and insert( )
Switch Statement	Another technique for **selection.** Similar to an IF statement.
For-Each Loop	An alternative to the traditional **For** loop. Iterates through each element in a collection. E.g. array

eBook Task Checklist

#	Basics	Sign Off	Mastered
1.0	Install JDK Compile and Run		
2.1	Output	HelloWorld.java	AdvancedOutput.java
2.2	Input (String)	Conversation.java	MixedRainbow.java
2.3	if Statement (String)	Password.java	ChooseMonth.java
2.4	Data Types & Exceptions	Album.java	PersonalData.java
2.5	**Dog in the Window**	Dog.java	Dog.java

#	Basics	Sign Off	Mastered
2.6	Calculations	SwimmingPool.java	RoundPool.java
2.7	Random Numbers	RandomDice.java	Blackjack.java
2.8	if Statement (Non-String)	ExamGrade.java	BiggestNumber.java
2.9	**Rock, Paper, Scissors**	RockPaperScissors.java	RockPaperScissors.java

#	Procedural	Sign Off	Mastered
3.1	Methods Calls	ModularDog.java	ModularDog.java
3.2	Validation	ModularValidation.java	ModularValidation.java
3.3	while Loop	ModularSafecracker.java	ModularSafecracker.java
3.4	for Loop	ModularNameChecker.java	ModularTenGreenBottles.java
3.5	**Fizzbuzz**	ModularFizzbuzz.java	ModularFizzbuzz.java

#	Procedural	Sign Off	Mastered
4.1	Creating Arrays	StarterArray.java	NumbersArray.java
4.2	Searching an Array	LinearNames.java	TenTimesBinary.java
4.3	Sorting an Array	InsertionLottery.java	BubbleLottery.java
4.4	2D & 3D Arrays	Minesweeper.java	Minesweeper.java
4.5	**Xando**	Xando.java	Xando.java

#	File Handling	Sign Off	Mastered
5.1	File Writer	TextToFile.java	TextToFile.java
5.2	File Reader	TextFromFile.java	TextFromFile.java
5.3	Encryption	SecretCaeser.java	SecretAtbash.java
5.4	**Cash Machine**	CashMachine.java	CashMachine.java

#	Final Challenges	Sign Off	Mastered
6.1	**Snakes and Ladders**	SnakesAndLadders.java	SnakesAndLadders.java
6.2	**Bingo**	Bingo.java	Bingo.java

#	External Classes	Sign Off	Mastered
7.1	Single Class *Practice Project*		
7.2	List Class *Practice Project*		
7.3	File Handling *Practice Project*		
7.4	Sorts & Searches *Practice Project*		
7.5	Building the GUI *Practice Project*		

#	Intro to GUI (eBook 2)	Sign Off	Mastered
8.1	Intro to GUI 1 *eBook 2*		
8.2	Intro to GUI 2 *eBook 2*		
8.3	Intro to GUI 3 *eBook 2*		
8.4	Intro to GUI 4 *eBook 2*		

Java Conventions

Conventions are rules which programmers follow to make their code more readable for themselves and other programmers. Below are several conventions you should attempt to adhere to:

Convention	Example

Class names	`public class FirstProgram`
	
	`//Starts with an uppercase letter`

Method names	`public static void main(String[] args)`
	
	`//Starts with a lowercase letter and ends with brackets`

Variable names	`int age;`
	
	`//Starts with a lowercase letter`

The Semicolon

A semicolon is used to end a statement.

Correct	Example
Output	`System.out.println("Hello World");`
Input	`name = inputScanner.nextLine();`
Calculation	`answer = number1+ number2;`
Method Call	`mp.changeName("Phil");`

DO NOT USE a semicolon in the following ways

Incorrect	Example
Class Declaration	`public class FirstProgram;`
Method Declaration	`public static void main(String[] args);`
IF statement	`if(iAge<18);`
WHILE loop	`while(bottles<100);`
FOR loop	`for(int i=0;i<100;i++);`