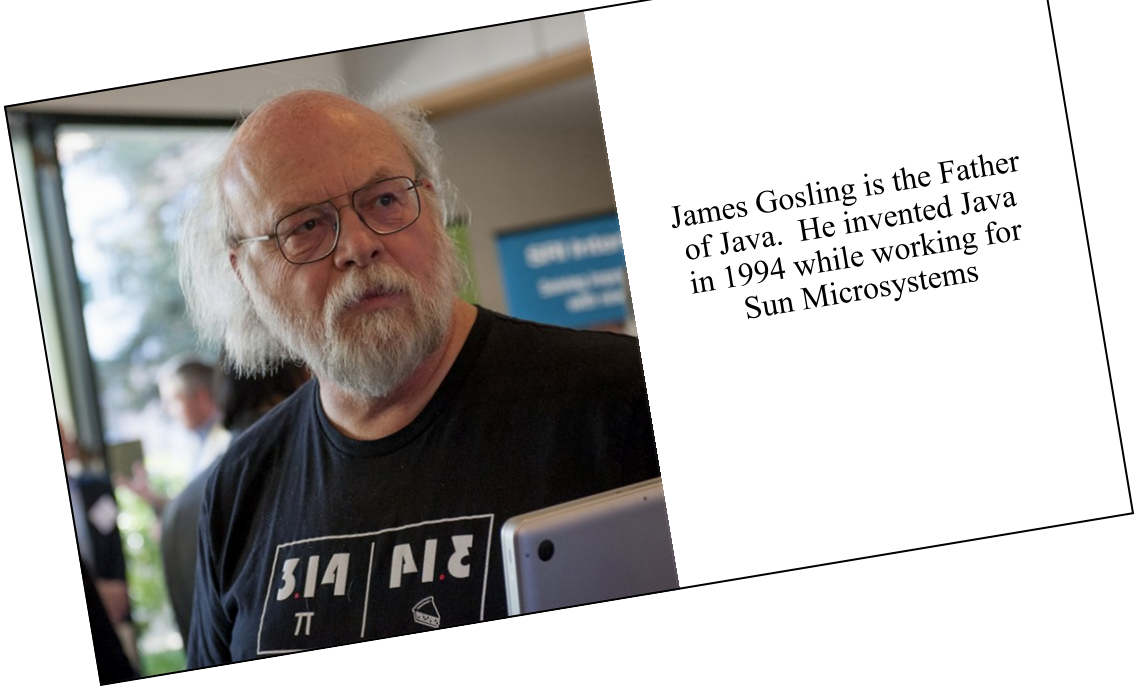
Contents



James Gosling is the Father of Java. He invented Java in 1994 while working for Sun Microsystems

Worksheets

Worksheet 1
Garage System

Reference Sheet

Writing Simple Classes

Classes and their properties

This is the class **Vehicle**. It has 4 properties as shown:

```
class Vehicle {
  String regNumber;
  int regYear;
  String model;
  String colour;
}
```

Inheritance

The class **Bus** extends **Vehicle**, and has an additional variable called *capacity*.
Therefore the class **Bus** looks like this:

```
class Bus extends Vehicle {
  int capacity;
}
```

Worksheet 1
Garage System

Activity Sheet

Here we will develop a simple application to handle vehicle details for a garage. One of the classes in this application is **Vehicle**. This class will have vehicle properties and vehicle-related methods.

Basic Level

1. Create the class **Vehicle** with the properties shown here:

```
class Vehicle {
  String regNumber;
  int regYear;
  String model;
  String colour;
}
```

2. Compile your class to see if there are any problems with your code.

Advanced Level

3. Add the following String variables to the class **Vehicle**:

   ownerID
   ownerName
   ownerTelNo

Expert Level

4. Create a new Class called **Car** that extends **Vehicle.**
5. Give your Class **Car** an integer property called *noOfSeatbelts*.
6. Create a new Class called **Motorcycle** that extends **Vehicle** and has another integer property called *noOfFreeHelmets*.

Worksheet 2
School System

Reference Sheet

Writing Simple Methods

Structure of a method

```
public void outputVehicle(){
   // body of method
}
```

Screen output

The following line outputs the contents of the property *name* for the object calling this method.

```
System.out.println (this.name);
```

Inheritance

The class **Bus** extends **Vehicle**, and has an additional variable called *capacity*. Therefore the class **Bus** looks like this:

```
class Bus extends Vehicle {
   int capacity;
}
```

Naming Conventions

- Variable names and method names should be in camelCase: e.g. outputVehicle
- Class names should start in a capital letter: e.g. Vehicle

Worksheet 2
School System

Activity Sheet

Here we will develop a simple application to handle a School System.
One of the classes in this application is **Person**.

Basic Level

1. Create class **Person** with Properties:
   indexnumber
   name
   surname
   address
   telephone number
   Decide which of these should be of type 'int' and which of type 'String'.

Advanced Level

2. Write a method called outputContactDetails() to output a person's name, surname, telephone number and address on the screen.

Expert Level

3. Create class **Student** that extends **Person** and also includes the properties:
   testMaths
   testEnglish
   testMaltese
   which will hold the test marks in the respective subjects.

4. In class **Student** create a method called outputStudent() to output student index number, name, surname and test marks on the screen.

Worksheet 3
Garage System

Reference Sheet

Writing Simple Methods

Structure of a method

```
public void outputPerson(){
   // body of method
}
```

The Main method

```
public static void main (String args[]) {
// body of method
}
```

The Constructor method

A Constructor method has the same name as the class and no return type.

```
public Student(){
        name = "No name entered";
        surname = "No surname entered");
}
```


Objects

Creating a new object

This line creates an object called studentA which is an instance of class Person

```
Person studentA= new Person();
```

Calling a method

This line creates calls the method outputPerson for studentA

```
studentA.outputPerson();
```

Worksheet 3
Garage System

Reference Sheet

Screen output

The following line outputs the contents of the property name for the object calling this method.

```
System.out.println (this.name);
```

Giving a property a value

The following line places 'Anna' in the property name of the object called studentA.

```
studentA.name = "Anna";
```

The Main Class

The main class in our application has a method called main. A main class can look something like this:

```
public class SchoolApp{
    public static void main (String args[]){
        Person studentA = new Person();

        studentA.name = "Joseph";
        studentA.surname = "Borg";
        studentAaddress = "7, Triq il-Linja, Lija";
        studentA.telNum = "21444444";

          studentA.displayObject();
        }
}
```

The Main Method

Creating object studentA of type **Person**. (**Person** is another class in this application)

Putting values in the properties of studentA

Calling the method displayObject for studentA

Worksheet 3
Garage System

Activity Sheet

Here we will continue developing the application we created in *Worksheet 1*.

We will create a method to output vehicle details and then create and work with an instance of a vehicle.

Basic Level

1. In class **Vehicle**, create a method outputVehicle() that will output the details of an object of class **Vehicle**.

Advanced Level

2. Create a class called **VehicleApp** that will be the main class of the application.

3. Create a main method in this class that creates an instance of the class **Vehicle** called 'vehicle1' and assigns its properties the following data:

regNumber	-	CBA 153
regYear	-	2010
model	-	Kia Venga
colour	-	white
ownerID	-	265788(m)
ownerName	-	Charles Borg
ownerTelNo	-	21 655 784

4. For your object 'vehicle1' call the method outputVehicle() in class **Vehicle** that outputs the details of your object.

Expert Level

5. Write a Constructor method for the Class **Vehicle** that initializes the instance variables to contents of your choice.
6. In method main, create a new instance of class **Vehicle** called 'vehicle2' and initialize it using the constructor method you wrote.
7. In the class **Car** create a method that will output the details of instances of the class. Call this method outputCar().
8. In your method Main create a new objects of type **Car** called 'car1' and assign it values of your choice from within method Main.
9. Call the method outputCar() for 'car1'.

Worksheet 4
Windows Order System Reference Sheet

Screen Output

The following line outputs the word 'Name:' followed by the contents of the property name for the object involved.

```
System.out.println ("Name:" + this.name);
```

Escape Characters

This table shows the codes that can represent special characters.

Example

Escape	Meaning
\n	New line
\t	Tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\ud	Unicode character


```
System.out.println (" Java! \n A really great programming language.");
```

Inline Comments

```
System.out.println (" Java!");
// Output 'Java' on the screen
```

Worksheet 4
Windows Order System

Activity Sheet

Here we will create a simple Windows Ordering System. We will first create a class called **Window** which will have a number of properties and methods. Then we will create objects of that class and manipulate them.

Revision

1. Create a class called **Window** having the following properties and methods:

   Properties:
   - *name*
   - *length*
   - *breadth*

   Methods:
   - showWindowDetails() [displays the name, length and breadth of a window]

2. Create a main class **WindowApp** with a main method in which you create a Window object called 'window1' and assign it the following data:
   - name = "Kitchen Window"
   - length = 2
   - breadth = 3

3. From the main method call the method showWindowDetails() for 'window1'.

Basic Level

4. When the length and breadth are displayed the output should look like this:
   Name:Kitchen Window
   Length:2m
   Breadth:3m

5. Use inline comments to explain your methods.

Advanced Level

6. Use the 'tab' literal to make the output look like this:

Name	:	Kitchen Window
Length	:	2m
Breadth	:	3m

Worksheet 5
Windows Order System Reference Sheet

Constants

Constants are declared as 'static' and 'final' variables

```
static final int AGE = 5;
```

Keyboard Input

Keyboard input using Scanner class

1. Import scanner utility

```
import java.util.Scanner;
```

2. Create a Scanner object

Type	Variable name	=	new	type	();
Scanner	input	=	new	Scanner	(System.in);

3. Read data from keyboard into variables

Reading an integer	int a = (input.nextInt());
Reading a double variable	double a = (input.nextDouble());
Reading a String variable	String a = (input.nextLine ());

Worksheet 5
Windows Order System

Activity Sheet

Here we will continue working on the Windows Ordering System we started in *Worksheet 4*. We will develop our application so that it will be able to output quotes on ordered windows.

Basic Level

1. Set a constant variable in class Window called *VATRATE* as 18% (0.18).
2. Now include customerName as an attribute of the class **Window**.
3. Create a method in class **Window** called newWindow() which reads window details (including customer name) from the keyboard.

Advanced Level

4. Edit your program so that when it outputs window details it also outputs:
   The customer name
   The VAT rate
5. Rewrite your main method so that it does the following:
   a. Create a new window called 'window1'.
   b. Call newWindow() in order to allow the user to input the information for the new window.
   c. Display the VAT rate and the window details.

Expert Level

6. Do the same for a second window called window2. Make sure that the output includes a title and skips a line between one window and the next.
7. Make your program run on a fresh (clear) screen every time.

Worksheet 6
Windows Order System Reference Sheet

Arithmetic Operators

Adding a and b and storing answer in c.	**c = a + b;**
Subtracting b from a and storing answer in c.	**c = a - b;**
Multiplying a and b and storing answer in c.	**c = a * b;**
Dividing a by b and storing answer in c.	**c = a / b;**

Methods that return a value

The following method is not void, it is int. This means the last line has to return an int (return area) to the method calling it.

```
public int getArea(){
        int area = this.length * this.breadth;
        return area;
}
```


Parameter Passing

Method Signature

```
Public String getDay(int dayNo){
```

Calling a method

```
today.getDay(dayNo);
Object.methodname(parameters);
```

Worksheet 6
Windows Order System

Activity Sheet

Here we will continue working on the Windows Ordering System we started in *Worksheet 4* and *5*. We will now develop our application so that it will be able to output quotes on ordered windows.

Basic Level

1. *The length and breadth of a window are not always a whole number*. Edit your application such that length and breadth can be numbers with a decimal fraction (e.g. 3.5).
2. In your class **Window** create the following methods:
   a. getArea (returns the area of the window passed to it).
   b. getPerimeter (returns the perimeter of the window passed to it).

Advanced Level

3. In your class **Window** create a method called getCost() that returns the cost of the Window (including glass and frame).
   a. Take cost of glass as a constant at €5 per metre squared.
   b. Take cost of frame as a constant at €10 per metre.
   c. Take cost of hinges etc at €10.

Expert Level

4. In class **Window** write a method called getVATCost() that calculates the cost (including VAT) of the window passed to it .
5. Use your main method to call the relevant methods in order to output the window details including: area, perimeter, total cost and cost including VAT.

Worksheet 7
Triangles Application

Reference Sheet

Unary Operators

Operator	This is…	…equivalent to	Operation
++	**n++**	**n = n + 1**	Add 1
--	**n--**	**n = n - 1**	Subtract 1

The Math Class

Importing the Math Class

Importing the Math class: **import static java.lang.Math.*;**

Method	Description
abs(int x)	Returns the absolute value of x
pow(int y, int x)	Returns y to the power of x.
sqrt(double x)	Returns the square root of x.

Worksheet 7
Triangles Application

Activity Sheet

Basic Level

1. Create a class called **Triangle** that has three properties: *a*, *b* and *hyp* which are the sides of a right-angled triangle.
2. In class **Triangle:**
   a. write a method newTriangle() that allows the user to input *a* and *b*.
   b. create a method called getHyp() that, given sides a and b, uses Pythagoras' theorem to find and return the hypotenuse (hyp).
   c. create a method called getArea() that, given a and b (the base and height), finds and returns the area of the triangle.
3. Create a main class called **TriangleApp** with a main method that creates an instance of Triangle called triangle1.
4. From the main method call the method newTriangle() for 'triangle1'.
5. From the main method call a method called outputDetails() in class Triangle that outputs the dimensions and area of 'triangle1'.
   (Note you have to write the method outputDetails() first).

Advanced Level

6. In class **Triangle** create a method called makeBigger() that uses the unary operator (++) to increase a and b by one and then outputs the new dimensions.
7. In class **Triangle** create a method called makeSmaller() that uses the unary operator (--) to decrease a and b by one and then outputs the new dimension.
8. Insert a call to these methods from your main method for your object 'triangle1'.

Expert Level

9. In class **Triangle** create a method getAverage() that returns the average length of the sides of a triangle passed to it.
10. Insert a call to this method from your main method to output the average length of the sides.

Worksheet 8
Number Game

Reference Sheet

Finding the Remainder

Modulus to find the remainder

Returns the remainder obtained after dividing a by b. This is called modulus.	**c = a % b;**

Generating a random number

Importing the Math Class

```
import static java.lang.Math.*;
```

Using the random method

random()	Returns a random number between 0 and 1.

The If-else statement:

```
if (mark == 100){
   System.out.println ("Well Done");
}
else {
  System.out.println ("Not full marks");
}
```

Worksheet 8
Number Game

Activity Sheet

Basic Level

1. Create a class called **NumberGame**.
2. Create a method called guessNumber that asks the user to guess a number. The method should output 'You Guessed!' if the user guesses and 'Wrong Guess!' if he doesn't.
3. Create a main class with a main method that creates an object of type numberGame called game1 and then calls the guessNumber() method.

Advanced Level

4. In the class **NumberGame** create another method called guessMultiple() that outputs a random number (between 2 and 10) on the screen and asks the user to input a multiple of it. The method should then output 'Correct Multiple' or 'Incorrect Multiple' accordingly.
5. In the main method call the method guessMultiple() for your NumberGame object.

Expert Level

6. In the class **NumberGame** edit the method called guessNumber() to generate a number, allow the user a guess and then output 'You guessed', 'Too large' or 'Too small' accordingly.

Worksheet 9
Bank Account Application

Reference Sheet

Finding the Remainder

Modulus to find the remainder

Returns the remainder obtained after dividing a by b. This is called modulus.	**c = a % b;**

Generating a random number

Importing the Math Class

```
import static java.lang.Math.*;
```

Using the random method

random()	Returns a random number between 0 and 1.

The If-else statement

```
if (mark == 100){
   System.out.println ("Well Done");
}
else {
  System.out.println ("Not full marks");
}
```

Worksheet 9
Bank Account Application

Activity Sheet

Basic Level

1. Create a class called **Account** with properties: userID, balance and interestRate. (Interest rate is likely to be a fixed amount)
2. Create a method called newAccount() that asks for the bank account details.
3. Create a main class **AccountApp** with a main method that:
   - Creates an account instance called *account1*.
   - Calls the newAccount() method for this account.
4. Create a method called makeDeposit() that adds the amount deposited to the balance.
5. Create a method called makeWithdrawal() that subtracts the amount withdrawn from the balance. However, if the resulting balance will be less than 0, your method should instead and output the message 'Insufficient funds available.'

Advanced Level

6. In class **Account** create a method called findSimpleInterest() that outputs the simple interest that would be obtained in a year on the present balance of a given account.
7. In method main call the findSimpleInterest() method for your account instance.

Expert Level

8. Create a method in Account called displayMenu() that asks the user whether he would like to deposit to or withdraw money from his account and then calls the appropriate method (makeDeposit() or makeWithdrawal()) accordingly.

Worksheet 10
Bank Account

Reference Sheet

Structure of a Case Statement

```
switch (expression){
        case 1: {
                  statement/s;
                  break;
                }
        case 2: {
                  statement/s;
                  break;
                }
        default: {
                   statement/s;
                  }
}
```


Using a Dialog Box

Importing swing

```
import javax.swing.JOptionPane;
```

Outputting a simple message dialog box:

```
JOptionPane.showMessageDialog(null, "Game Over");
```

Worksheet 10
Bank Account Application

Activity Sheet

Here we will continue working on the Bank Account Application we started in *Worksheet 9* this time including a Menu of Options and some elements of GUI.

Basic Level

1. Edit your method displayMenu() to output a menu with the following options:
    1. New Account
    2. Deposit to Account
    3. Withdraw from Account
    4. Calculate Simple Interest
    5. Exit
2. Implement a Case statement to edit your application such that when it runs it displays the Main Menu and then takes you to the appropriate method according to the user's choice.

Advanced Level

3. Create a method called exitApplication() that will be called when the 'Exit' option is chosen and will display 'Quitting Application' on the screen.
4. Edit your 'switch' construct such that if the menu choice entered is not 1-4 the message 'Invalid choice' is output.

Expert Level

5. Use a JOptionPane command to use a Text Box in order to display the 'Quitting Application' message.

Worksheet 11
Bank Account

Reference Sheet

Structure of a do..while loop

```
do {
   System.out.println ("MENU");
   System.out.println ("1. Enter Students Details");
   System.out.println ("2. View Student Details");
   System.out.println ("3. Exit");
   System.out.print   ("Enter choice:");
  choice = (input.nextInt());
} while (choice!=3);
```


Using an Input Dialog Box

```
import javax.swing.JOptionPane;
```

```
choices = JOptionPane.showInputDialog(
                    "1. Try Test\n"+
                    "2. Get Grade and Rank\n"+
                    "3. Quit\n"+
                    "Enter choice: \n");
          choice = Integer.parseInt(choices);
```

Worksheet 11
Bank Account Application Activity Sheet

Here we will continue working on the Bank Account Application we started in *Worksheets 9 and 10.* This time we will be including a Menu of Options that is repeatedly displayed until the user chooses to quit the application. We will also include more elements of GUI.

Basic Level

1. Edit the method displayMenu() of your Bank Application so that the program loops until the user selects (4) the exit option.

Advanced Level

2. Create a constructor method for the **Account** class. This method should initialize balance to 0.00 and userID to '1'.
3. Implement a method called 'freshAccount' in your class Account that gives the user the choice between the default settings for a new Account or entering his own details. Make sure the user is given this option when he chooses to start a new account.
4. Make your application password protected: The displayMenu() method should not display the menu until the user enters the correct password.
   Use a constant to store the password.

Expert Level

5. Edit your displayMenu() method such that the menu is displayed in a dialog box in which the user can then enter his menu option.

Worksheet 12
Marks System

Inheritance

```
Public class Cat extends Pet{

}
```

For..Loop

for	(	initialization	;	condition	;	iteration	)
for	(	i = 0	;	i<10	;	i++	)

```
for (x=1; x<=10; x++) {
    System.out.println("x=" +x);
 }
```

Worksheet 12
Marks System

Here we will create a system to handle exam marks.

Basic Level

1. Create class **Person** with attributes name, surname
2. Create class **Student** that extends **Person** with attributes class, totalExamMark, indexNumber
3. Create, in class **Student**, a method called enterStudent() that allows the user to enter the details of a student.

   To obtain the total exam mark use a for loop that accepts 10 marks and finds their total.
4. Create a main class called MarksApp and in the main method create an instance of Student called student1 and call the method enterStudent() for this student.

Advanced Level

5. Create a method getStatistics() that returns the average mark for a student.
6. Create a method called outputDetails that outputs student details, including the average mark.

Expert Level

7. In class **Student** Create a method called mainMenu() that will display a menu with the following options in a Dialog box:

   1. Enter Student
   2. View Student Details
   3. View Class Statistics
   4. Exit
8. Call mainMenu() from the main method: mainMenu() should loop until the user selects Option 4.
9. Implement a Switch Statement for the main menu and in case 1 and 2 it should take the user to the relevant method.

Worksheet 13
Marks System

Reference Sheet
Activity Sheet

Here we will continue building a system to handle exam marks.

Declaring and assigning an array

array-variable		=	new	type	[size];
int[ ]	marks	=	new	int	[5];

For..Loop

for	(	initialization	;	condition	;	iteration	)
for	(	i = 0	;	i<10	;	i++	)

```
for (x=1; x<=10; x++) {
    System.out.println("x=" +x);
 }
```


Basic Level

1. Edit your Application such that the students' marks are entered in an array. You will therefore need to add an array of marks to your instance variables.

   You will also need to edit some of your methods.

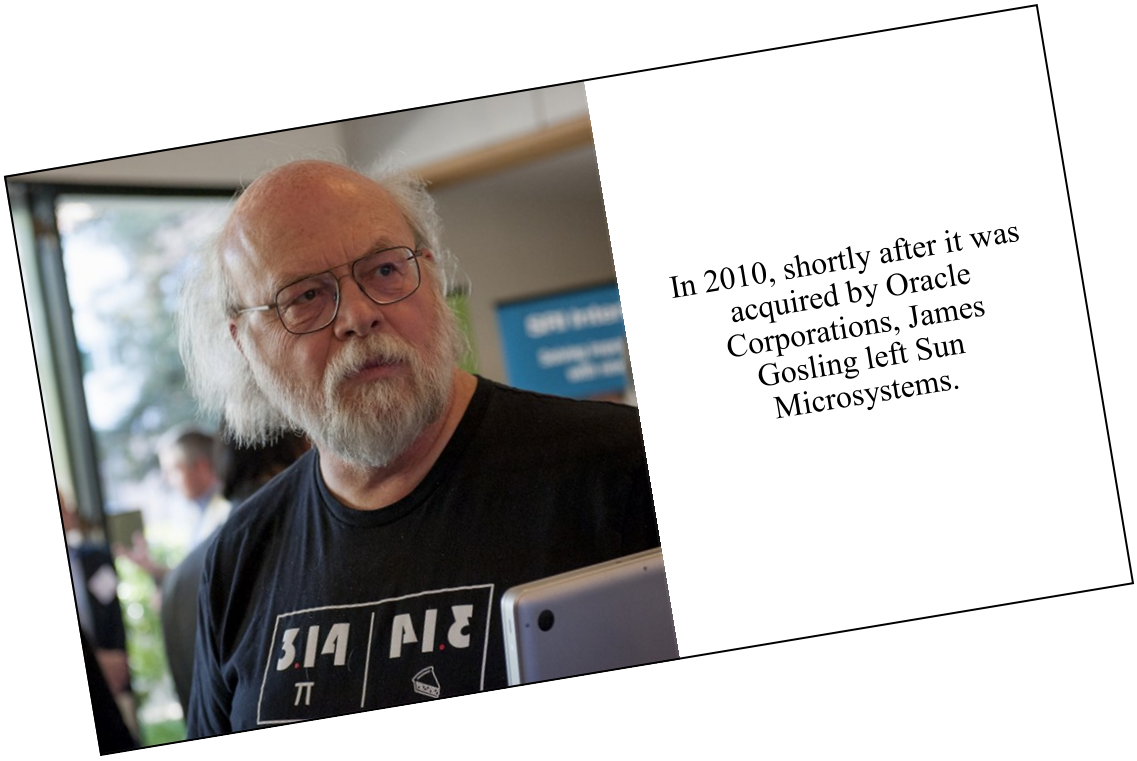
2. Edit your method getStatistics() such that it outputs the Average, Highest and Lowest mark of each student.

Advanced Level

3. Create a class called **Group** that will handle 15 students. Create a method in it called enterStudents() that reads student details.

Expert Level

4. In class **Group**, create your method viewClassStats () that will output: highest, lowest and average total exam marks of the class.
5. In class **Group**, create your method viewStudentDetails () that will output the details of the student with a given index number.


In 2010, shortly after it was acquired by Oracle Corporations, James Gosling left Sun Microsystems.

Revision

Revision 1 - 5

Boxes

Activity Sheet

Basic Level

1. Create a class called **Box** that has the following properties:
   - *name*
   - *length*
   - *breadth*
   - *height*

Advanced Level

3. Create the following methods for **Box**:
   - outputBox() - This method should output the details of the box
   - enterBox() - This method should ask the user for the length, breadth and height of the box
   - getVolume() - This method should calculate and output the volume of the box
4. Create a class called **BoxApp** that includes the main method.
5. Your main method should:
   - Create a box object called 'boxA'.
   - Call the method enterBox for 'boxA'.
   - Call the method outputDetails() for 'boxA'.
   - Call the method getVolume() for 'boxA'.
   - Do the same for a second Box called 'boxB'.

Expert Level

6. In class **Box** create a method called outputCost() that calculates and displays the cost of making a solid box the size of a given box. (The material is €5 per cubic metre.)
7. Call the method outputCost() from your main method for your objects 'boxA' and 'boxB'.

Ultimate challenge:

8. The buyer is charged 18% VAT on each box bought. In class **Box** create a method called findVATCost() that finds and outputs the cost including VAT of a given box .
9. Call the method findVATCost() from your main method for your objects 'boxA' and 'boxB'.

Revision: Worksheets 1 - 6
Books

Activity Sheet

Basic Level

1. Create a class called **Book** that includes the following properties:

   (Give each property a reasonable name and type)

   - Subject
   - Author
   - No. of pages
   - Cost

Advanced Level

2. Include the following methods in your class **Book**:

   - newBook() (asks the user to enter book details)
   - showBook() (outputs book details)
   - getVATCost() (finds and RETURNS the book cost including VAT)
   - getDiscountCost() (finds and RETURNS the cost of book at 20% discount)

Expert Level

3. Create a class called **BookApp** that includes the main method written so as your application does the following:

   a. Create an object called 'whiteBook'

   b. Enter details for this book (call relevant method)

   c. Show details for this book (call relevant method)

   d. Show the Cost Including VAT of this book

   e. Show the discounted price of this book

4. Do the same for the following two other objects of type Book:

   - 'greenBook'
   - 'orangeBook'

Revision: Worksheet 1 - 6 Shapes

Activity Sheet

Basic Level

1. Create a class called **Quadrilateral** that has the following properties:
   - *sideA*
   - *sideB*
2. Create a class called **Parallelogram** that extends **Quadrilateral** and also includes the following property:
   - *perpendicularHeight* (where side A is the base)

Advanced Level

3. Create the following methods for **Parallelogram**:
   - getArea()
     This method should calculate and return the area of the parallelogram
   - getPerimeter()
     This method should calculate and return the perimeter of the parallelogram
   - outputDetails()
     This method should output Parallelogram details in this format:

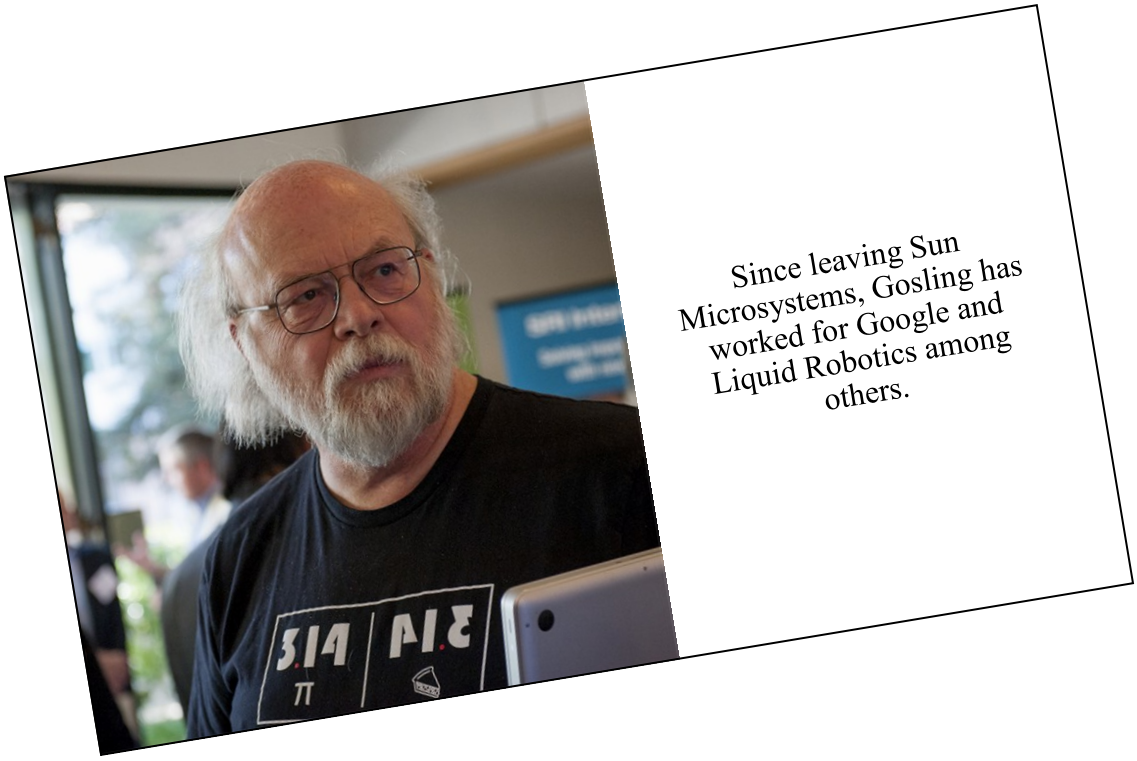
Side A	:	5 cm
Side B	:	6 cm
Perpendicular height	:	5 cm
Area	:	25 sq cm
Perimeter	:	22cm

Expert Level

4. Create a class called **QuadrilateralApp** that includes the main method.
5. Your main method should:
   - Create a parallelogram object called 'parallelogram1' and assign it values from within your method.
   - Call the method outputDetails() for 'parallelogram1'.
   - Do the same for a second parallelogram called 'parallelogram2'.

Ultimate challenge:

6. In class **Parallelogram** create a method called printCost() that calculates the cost of making a rug the size of a given parallelogram and returns it to the method calling it. The material is 5 Euro per square metre.
7. The above should be output through the method outputDetails().


Since leaving Sun Microsystems, Gosling has worked for Google and Liquid Robotics among others.

What application would you like to create?

Introduction

Taken from *State Schools Computing Syllabus*

This practical component completes the Programming exercise as requested by the MATSEC board. Also, it carries 15% of the Form 5 Annual Examination assessment. Each student has to produce his/her own work.

Marks are awarded for the following sections in the coursework.

- Twenty six marks out of the 30 marks are allotted for:
  - Definition of the Problem
  - Solution of the Problem
  - Running the program
  - User instructions
  - Comments and conclusions
- The other 4 marks are to be awarded for the overall layout and presentation of the coursework.

More details may be sought from the official SEC syllabus.

This global practical mark for the SEC examination shall also be used to satisfy the 15% set aside for the practical component of the Form 5 annual examination mark. Obviously the mark has to be divided by 2. With this in mind, it is imperative that the practical component be finalised by the end of the calendar year so that the teacher has ample time to finalise his/her corrections before the commencement of the Form 5 annual examination session in February.

Moderation of coursework at Form 5 may be exercised by the Computing Department and/or MATSEC personnel.

SEC Coursework Marking Scheme

IMPORTANT: For moderation purposes by MATSEC, this completed sheet is to be available together with the coursework exercise.

Criteria for Assessment	Maximum Mark	Actual Mark
A: PROGRAMMING		
Definition of the problem:		
Description of the scope of the problem to be tackled	**2**	
Statement of the results required	**2**	
Details of the input information required	**2**	
Solution of the problem:		
Algorithm (flowchart or pseudocode)	**4**	
Computer listing of program	**4**	
Details of any special design features	**4**	
Running the program		
Evidence that the solution works	**2**	
Plan of test data	**2**	
User instructions:		
Loading and operating the program	**2**	
Comments and conclusion:		
Limitations and improvements	**2**	
TOTAL MAXIMUM FOR PROGRAMMING	**26**	
B: LAYOUT AND PRESENTATION		
Clear indication of each section mentioned above	**1**	
Use of headers and footers (to include page numbers)	**1**	
Use of tables	**1**	
Use of bulleted and/or numbered lists	**1**	
TOTAL MAXIMUM FOR LAYOUT/PRESENTATION	**4**	
TOTAL	**30**	

Guidelines for Awarding Marks:

Sections with a maximum mark of 2:

Award marks as follows

0 – not attempted;

1 – partially presented;

2 – fully correct response.

Algorithm: award 4 marks for a complete, readable, understandable, language-independent and easy-to-follow algorithm

Computer listing: award 4 marks for programs showing meaningful identifiers, correct indentation, use of correct coding rules and inclusion of comments

Special design features: award 2 marks for an adequate description of any special features included in the program (good user interface, use of advanced constructs and algorithms, etc). The other 2 marks are to be awarded for a description (and the eventual use) of simple objects.

Student's Name in blocks:

Tutor's Information:

Name in blocks:

Signature:

Date:

Definition of the Problem

Description of the Problem to be tackled **[2 marks]**

Introduction

Explain:

- What your program will do;
- Where it can be used;
- By whom it will be used.

Introduction

List the options in the Main Menu

1. _____
2. _____
3. _____
4. _____
5. _____

Advantages of Computerising the System

1. _____
2. _____
3. _____

Choice of Java as a Programming Language

1. _____
2. _____
3. _____

Definition of the Problem

Statement of the Results Required **[2 marks]**

Information output by this variable	Variable name	Variable type

Details of the Input Information required **[2 marks]**

Information input by this variable	Variable name	Variable type

Algorithm (flowchart or pseudocode)

General System flowchart **[4 marks]**

Include also flowcharts (or pseudocode) of all major methods in your application.

Solution of the Problem

List the classes this application will require

1. _____
2. _____
3. _____
4. _____

List the Properties and Methods of each of your classes

Worksheet
Marks System

Activity Sheet

You may create an application that handles students' details and marks and outputs statistics about those marks...or adapt the same idea to other records (e.g. Athletes).

1. Create class Student with the following attributes:
   name, surname, index number, year of birth and array of ten marks
   (add 2 or more further attributes)

2. Create the following methods:
   a. *Enter student*
   b. *View student*
   (this method may also output 'Pass' or 'Fail' for each mark entered)
   (this method may implement GUI elements for the output)
   c. *Get statistics*
   (options: this method may output highest, lowest and average marks)

3. Create method getAge() that calculates the students' age and returns it to the method viewStudent().

4. Create a Main Menu with the following options:
   1.Enter Student
   2.View Student
   3.Get Student Statistics
   4.Exit

5. The Main Menu should loop until the option 'Exit'is chosen.

6. Create a main class with a main method in which:
   a.An object of Student is created
   b.The method mainMenu() is called for your Student object.

7. Introduce a GUI element in your program.

Worksheet
CAL Quiz

Activity Sheet

Basic Level

Part 1: Create Your Classes

1. Create a class called **Question** with properties *quest*, *answer*, *correct* (*correct* should be of type Boolean)
   (You may also include *explanation* and *userAnswer*).
2. Create a class called **Test** with properties *testTopic*, *questionList* (an array of 10 or more questions) and *mark* which will hold the user's test mark.
   (You may also include *studentName*).
3. Create a class called **TestApp**.

Part 2: Create Your Methods

4. In class **Question**, create a method called displayQuestion()
   - This displays the question and answer and then outputs 'Correct' (if *correct* is true) or 'Wrong' (if *correct* is false).
   - You might consider using GUI elements here.
5. Create a constructor method for **Question** that initialises *correct* to 'false'.
6. Create a constructor for **Test** that sets *testTopic* to your chosen title and fills the array *QuestionList* with your questions (*quest*) and their answers (*answer*).
   - You should have 10 or more questions.
7. Asking the test questions:
   - In class **Test**, create a method called tryTest() that asks the user the test questions one by one and increments *mark* for each correct answer. (Call the method askQuestion() as needed)
8. In class **Test** create a method called readNotes() that outputs notes related to the test topic.
9. In class **Test** create a method called showTest() that outputs the questions with their correct answers as well as whether the user got them right or not.

Part 3: Create Your Main Method

10. Create your main class, **TestApp**, with a main method that creates an instance of Test and calls the methods readNotes() and tryTest() and finally displays the user's mark using GUI elements.

Let's go further! Need ideas on how to develop your application further? Here are some options.

Advanced Level

11. Edit your application so that the main method creates an instance of Test and gives the following Menu Options by calling a method displayMenu() in class **Test:**
    1. Read Notes
    2. Try Test
    3. Show Test
    4. Exit
12. Implement a switch to make your menu work and implement a loop structure so that the menu will loop repeatedly until the exit option is chosen.
13. If the user enters a number other than 1 to 4 the system should output: "Invalid Menu Choice" and then display the menu again.
14. Edit your program such that your Main Menu is displayed using GUI features.
15. Edit your program such that the message 'Invalid Menu Choice' is displayed using a GUI.

    (Consider implementing GUI features in other sections of your work.)

Expert Level

16. In class **Test** create a method called viewResults() that outputs the user's result, a performance comment and perhaps a grade. Edit your main menu accordingly.
17. The method showTest() should only be accessed if the user has first tried the test. Implement this restriction so that (if the user has already tried the test) it displays the test questions with their answer and for each question outputs 'Your answer was CORRECT/WRONG' accordingly. BUT if the user hadn't first tried the test the system should instead output the message 'Try the test first'.

    (Hint! A Boolean variable might be useful here).
18. In Class **Test**, create a method called makeTest() that allows the user to enter his own test questions. These questions should be stored into tQuestionList, an array of Question objects. The method could then save the questions to a text file.
19. Implement a password into method makeTest() so that only someone who has the password can enter test questions.
20. In Class **TestApp** create a method called chooseTest() that allows the user to choose between the teacher's test and the default test.

    (Else consider having more than one test available and letting the user choose which test to do)
21. Introduce the option 'Change Test Questions' into your Main Menu and make your Menu Option '1.Try Test' now lead to 'Choose Test', a menu that allows the user to choose between: the teacher's test and the default test.