

Tabish Parkar

Formative Assessment

1: Systems

Development 1

(HSYD100-1)

1.1. Class – Car.

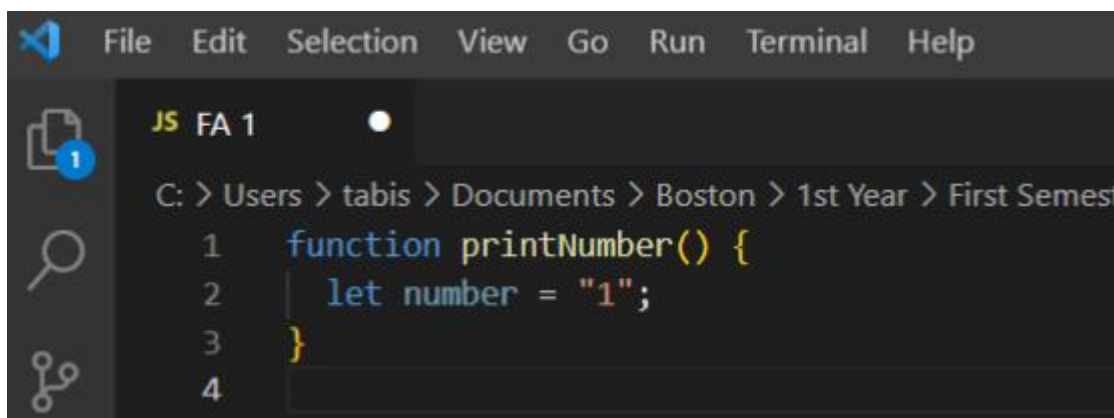
Objects – Doors, sunroof, semi-bullet proof windows and six wheels.

Method – The ability to go into sports mode.

1.2. Polymorphism is the ability to change the attributes of an object for example a car can be changed from eco mode to sport plus mode with a press of a button telling the engine control unit to release the engines full potential for the driver to use. The driver can then revert to eco mode to save fuel which is another form of polymorphism.

1.3. Developers use information hiding by encapsulation. Encapsulation is used in Object Orientated Programming to hide how specific methods work because it is not necessary to have more information about the method which would make the code more complex.

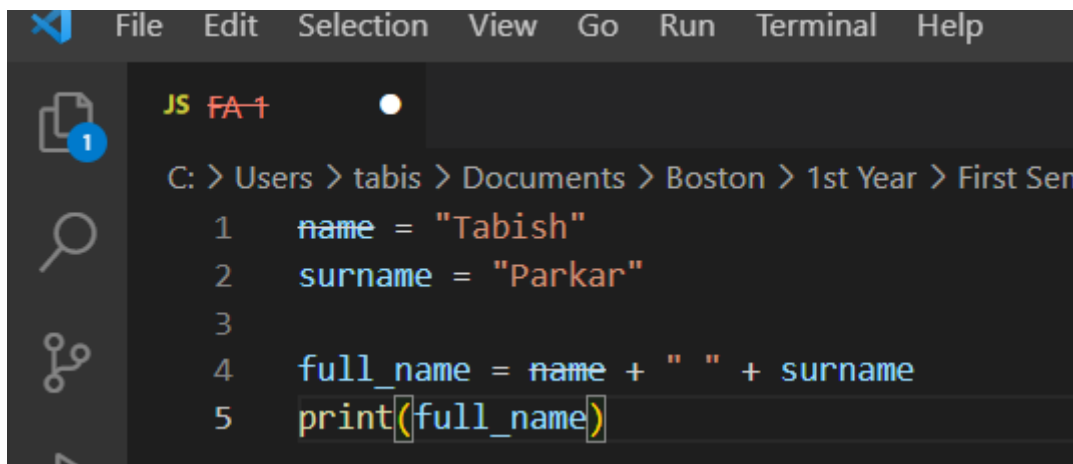
2.1. It refers to the availability of a variable within a block of code.

A screenshot of a code editor interface. The top menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The left sidebar shows icons for Explorer, Search, and Source Control. The main editor area has a tab labeled 'JS FA 1'. The file path is 'C: > Users > tabis > Documents > Boston > 1st Year > First Semest'. The code is as follows:

```
1 function printNumber() {  
2   let number = "1";  
3 }  
4
```

In the example above there is nothing in the block of code which will lead to an error.

2.2. This would be called concatenation.

A screenshot of a code editor window with a dark theme. The menu bar at the top includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The file explorer on the left shows a folder icon with a blue circle containing the number '1'. The code editor displays a JavaScript file named 'JS FA-1'. The file path is 'C: > Users > tabis > Documents > Boston > 1st Year > First Sen'. The code consists of five lines: 1. 'name = "Tabish"', 2. 'surname = "Parkar"', 3. (empty line), 4. 'full_name = name + " " + surname', and 5. 'print(full_name)'. The code is syntax-highlighted with blue for keywords, orange for strings, and yellow for function names and brackets.

2.3. Operator precedence is when a math sum is calculated in an ordered way like the BODMAS method. Multiplication, division and remainder operators have an equal precedence compared to addition and subtraction which have a lower precedence.

In this case the sum inside the bracket would be given precedence first. Then the sum of $7 * 2$ inside the second bracket which would equal 14. The original bracket remains with the sum now showing $4 * (5 + 14)$. $5 + 14 = 19$.

Lastly, we multiply $4 * 19$ which will yield us with the final answer of 76.

2.4. A named constant which can also be called a symbolic constant, is similar to a variable in the way that it has a name, data type and a value. A literal constant is when the value is taken literally. A numeric constant is in opposition to the character or string constant, it displays values.

3.1. When a return statement is embedded in the method, when an exception is put in a method and is not found inside the method and when a programmer codes a conditional statement which will terminate depending on the conditions.

3.2. Overloading is a method which gives you the ability to use one identifier to execute various objectives, to be more precise it means to write multiple methods in the same scope that has the same name but different parameter lists. When various methods share a name,

the compiler will understand which one to use dependent on the arguments in the methods call.

3.3. The section where the data items scope is visible to a program and it can also be referred to by using its simple identifier. A variable will be in scope from when it is stated lasting till the ending of the block of code within where the declaration lies.

3.4. Java method headers must contain a return type, an identifier, optional access specifiers, parentheses which may or may not be empty and an optional static modifier.

4.1. The beginning of the main method.

4.2. Creates a scanner where a user can input characters.

4.3. User must enter their first name which will be assigned to the string.

4.4. Shows a greeting using the users first name. There is also a parameter.

4.5. Variables

4.6. F

4.7. There is no return value.

4.8. Void return type

4.9. Shows a string which will include the first name of the user along with the greeting in line
G

```
1  public Employee()  
2  
3  public Employee(String n, String sn, String num) {  
4      empName = n;  
5      empSurname = sn;  
6      empNumber = num;  
7  }  
8  
9  public void setEmpName(String nm) {  
10     empName = nm;  
11 }  
12  
13 public void setEmpSurname(String snm) {  
14     empSurname = snm;  
15 }  
16  
17 public void setEmpNumber(String num) {  
18     empNumber = num;  
19 }  
20  
21 public void setEmpSalary(double sal)  
22     empSalary = sal;  
23  
24  
25 public String getEmpName() {  
26     return empName;  
27 }  
28  
29 public String getEmpSurname() {  
30     return empSurname;  
31 }  
32  
33 public String getEmpNumber() {  
34     return empNumber;  
35 }  
36  
37 public double getEmpSalary() {  
38     return empSalary;  
39 }  
40
```

```
40
41 public void increaseSalary(double sal) {
42     empSalary += amt;
43 }
44
45 public String toString() {
46     return "Employee Name: " + empName + "\nEmployee Surname: " + empSurname + "\nEmployee Number: " + empNumber + "\nEmployee Salary: " + empSalary
47 }
```

```
1  Scanner input = new Scanner(System.in);
2
3  //Creating first employee object using no-argument constructor
4  Employee emp1 = new Employee();
5
6  System.out.print("Enter employee name: ");
7  emp1.setEmpName(input.nextLine());
8
9  System.out.print("Enter employee surname: ");
10 emp1.setEmpSurname(input.nextLine());
11
12 System.out.print("Enter employee number: ");
13 emp1.setEmpNumber(input.nextLine());
14
15 System.out.print("Enter employee salary: ");
16 emp1.setEmpSalary(input.nextDouble());
17
18 input.nextLine(); //Consume the new line character
19
20 //Creating second employee object using parameterized constructor
21 System.out.println("\nCreating second employee object\n");
22 System.out.print("Enter employee name: ");
23 String empName = input.nextLine();
24
25 System.out.print("Enter employee surname: ");
26 String empSurname = input.nextLine();
27
28 System.out.print("Enter employee number: ");
29 String empNumber = input.nextLine();
30
31 System.out.print("Enter employee salary: ");
32 double empSalary = input.nextDouble();
33
34 Employee emp2 = new Employee(empName,empSurname,empNumber);
35 emp2.setEmpSalary(empSalary);
36
37 //Printing details of both employees
38 System.out.println("\nDetails of Employee 1:\n" + emp1.toString());
39 System.out.println("\nDetails of Employee 2:\n" + emp2.toString());
40
41 //Increasing the salary of both employees by 5%
```

```
41 //Increasing the salary of both employees by 5%
42 emp1.increaseSalary(emp1.getEmpSalary() * 0.05);
43 emp2.increaseSalary(emp2.getEmpSalary() * 0.05);
44
45 //Printing details of both employees after salary increment
46 System.out.println("\nDetails of Employee 1 after salary increment:\n" + emp1.toString());
47 System.out.println("\nDetails of Employee 2 after salary increment:\n" + emp2.toString());
48
49 input.close(); //Closing scanner object
```

```
1 //Increasing the salary of both employees by 10% and 15%
2 emp1.setEmpSalary(emp1.getEmpSalary() * 0.1);
3 emp2.setEmpSalary(emp2.getEmpSalary() * 0.15);
```

```
1 //Printing details of both employees after salary increment
2 System.out.println("\nDetails of Employee 1 after further salary increment\n" + emp1.toString())
3 System.out.println("\nDetails of Employee 2 after further salary increment\n" + emp2.toString());
```