



Topic 10 – Introduction to Object Orientated Programming

Question 1

Create a Java application that contains two separate classes:

- Pizza
- Main

The Pizza class will model a Pizza object! Create the following attributes (instance variables) and methods (actions / behaviours) of a Pizza.

Attributes (Instance Variables)

- A pizza is of a specific size (small, medium, large). A pizza can have toppings such as cheese and mushrooms. A pizza has a diameter measured in inches, such as 14.00. A pizza has a price such as €12.
- Create instance variables as per the above underlined text.

Methods (actions / behaviours)

- Create the following methods to represent the actions / behaviours of a Pizza object.
- A method named **eat()**, which returns the message, "Eat a pizza slowly, it can choke you if you eat it too fast!" The return type of the method should be String and not void. Use the return keyword to return the message.
- A method named **toString()**, which returns information (a string) about a given instance (an object) created from the Pizza class. Here is the code for this method, as it has not been previously discussed on the course. The "\n" character starts a new line.

```
public String toString(){
    return "Size: " + size + "\n" + "Toppings: " + toppings + "\n" + "Diameter: " + diameter +
    "\n" + "Price: " + price;
} // method
```

In the second class of the application (Main), create the main method.

In the main method, create three Pizza objects with the following attributes:

Size	Toppings	Diameter	Price
Small	Ham and Pineapple	12.00	€10
Medium	Chicken and Mushroom	14.00	€12
Large	Tuna and Sweetcorn	16.00	€14

For each object, call the **toString()** and **eat()** methods in turn.

Store your source code in a folder named **JFT10Ex1**



```
Ex1>java Main
Size: Small
Toppings: Ham and Pineapple
Diameter: 12.0
Price: 10.0
Eat a pizza slowly, it can choke you if you eat it too fast!

Size: Medium
Toppings: Chicken and Mushroom
Diameter: 14.0
Price: 12.0
Eat a pizza slowly, it can choke you if you eat it too fast!

Size: Large
Toppings: Tuna and Sweetcorn
Diameter: 16.0
Price: 14.0
Eat a pizza slowly, it can choke you if you eat it too fast!
```

Question 2

Create a Java application to model a **Jelly** class. Jelly objects will be instantiated (created) from the Jelly class.

A Jelly should have the following **attributes**:

manufacturer	String
price	float
flavour	String
noCalories	float
expiryDate	String

A Jelly should have the following **methods**:

- **setInstructions()** – a method that returns the message, “Leave Jelly to set in a cool environment for 1hr”.
- **toString()** – a method that returns a string, capturing the values held in the instance variables / attributes of a Jelly object, created from the Jelly class.

In the second class of the application (Main), create the main method.

In the main method, create three Jelly objects with the following attributes:

manufacturer	price	flavour	noCalories	expiryDate
Chivery	3.50	Strawberry	24	25/06/2016
Nestles	3.42	Banana	26	27/07/2016
Danones	3.00	Lemon	28	28/06/2015

For each object, call the **setInstructions()** and **toString()** methods in turn.

Store your source code in a folder named **JFT10Ex2**



```
Ex2>java Main
Manufacturer: Chivery
Price: 3.5
Flavour: Strawberry
No. of Calories: 24.0
Expiry Date: 25/06/2016
Leave Jelly to set in a cool environment for 1hr

Manufacturer: Nestles
Price: 3.42
Flavour: Banana
No. of Calories: 26.0
Expiry Date: 27/07/2016
Leave Jelly to set in a cool environment for 1hr

Manufacturer: Danones
Price: 3.0
Flavour: Lemon
No. of Calories: 28.0
Expiry Date: 28/06/2015
Leave Jelly to set in a cool environment for 1hr
```

Question 3

I would like you to review the following program and answer the questions as presented using pen and paper.

Then, to determine whether you have the correct answers, create the program in Notepad++.

Store your source code in a folder named **JFT10Ex3**

```
public class Dog{

    public String name;
    public byte age;
    public String breed;

    public String makeNoise(){
        return "Woof....";
    }// method

    public String toString(){
        return "Name: " + name + "\n" + "Age: " + age + "\n" + "Breed: " + breed;
    }// method

} // class
```

Please turn over



```
public class Test{

    public static void main(String[] args){

        Dog d1 = new Dog();
        d1.name = "Frodo";
        d1.age = 4;
        d1.breed = "Alsatian";

        Dog d2 = new Dog();
        d2.name = "Labby";
        d2.age = 3;
        d2.breed = "Labrador";

        Dog d3 = d2;
        d3.age = 6;

        System.out.println(d1.name); // Answer?
        System.out.println(d2.age); // Answer?
        System.out.println(d3.age); // Answer?

        System.out.println(d1.age == d2.age); // Answer?
        System.out.println(d1 == d2); // Answer?
        System.out.println(d1 == d3); // Answer?
        System.out.println(d2 == d3); // Answer?

    } // main

} // class
```

Question 4

Create a Java class named Numbers to perform simple mathematical operations. Include the following methods:

Method Signatures

public int sum(int num1, int num2)
public float divide(float num1, float num2)
public float multiply(int num1, int num2)
public float remainder(float num1, float num2)
public void message() (print the String: This is a class that performs simple, mathematical operations)

Please turn over



```
public class Test{
    public static void main(String[] args){
        Numbers n1 = new Numbers();
        System.out.println("Sum: " + n1.sum(5,10));
        System.out.println("Divide: " + n1.divide(10,10));
        System.out.println("Multiply: " + n1.multiply(15,10));
        System.out.println("Remainder: " + n1.remainder(100,90));
        n1.message();
    } // main
} // class
```

```
emos\Numbers>java Test
Sum: 15
Divide: 1.0
Multiply: 150.0
Remainder: 10.0
This is a class that performs simple, mathematical operations.
```

Store your source code in a folder named **JFT10Ex4**

Question 5

Write a Java program to determine whether an integer falls within a range of values.

Create a class named **NumberRange** and include a method named **checkInRange**. The method should be passed three values (**int** lower_range_value, **int** upper_range_value and **int** no_to_search). If the no_to_search is found within the range, the message, "No. in range" should be returned. Otherwise, the message, "No. not in range!" should be returned.

Create a **Test** class and create a NumberRange object in the main method. Call the checkInRange() method and pass the following values, (5,500,333). Output the value of the String returned from the method.

Store your source code in a folder named **JFT10Ex5**

```
No. in range
```

Please turn over



Question 6

Write a Java program to determine the number of vowels found in a piece of text.

Create a class named **VowelCount**.

The class should include a method also named vowelCount(). One method parameter should be specified, a String named message.

The method should return a String detailing the total number of vowels and breakdown (a, e, i, o, u) found in the message.

Create a second class named **Test**. In this class create an object of type VowelCount from within the main method. Call the method, vowelCount() and pass the String, "Walking on the moon", to the method.

- **Hint:** Investigate the String class in the API to find the following methods:
 - A method to return the length of a String.
 - A method to extract a character at a specific index of a String.

Store your source code in a folder named **JFT10Ex6**

```
No. of vowels: 6
a count: 1
e count: 1
i count: 1
o count: 3
u count: 0
```

Question 7

Write a Java class named **Sum** and include the following overloaded methods. The methods should return the sum of the values passed.

public int sumTwoValues(int num1, int num2)
public float sumTwoValues(float num1, float num2)
public double sumTwoValues(double num1, double num2)
public int sumTwoValues(byte num1, byte num2)
public int sumTwoValues(byte num1, int num2)

Create a second class named **Test** and include the main method. Create an object of class type Sum and call each of the overloaded methods in turn, passing appropriate values. For example:

- System.out.println(s1.sumTwoValues(4,5)); // call the int method version

Please turn over



```
opic 10 - Exercises\JFT10Ex7>>java Test
9
9.0
9.0
9
9
```

Store your source code in a folder named **JFT10Ex7**

Question 8

Write a Java class to model a **Student** (Student.java).

The class should have the following instance variables.

```
public String firstName;
public String lastName;
public byte age;
public String className;
public float gradeAverage;
public String address;
```

The class should also have the following instance methods.

- toString()

Create a second class named **Test**.java. In the main method, create three Student objects with the following attributes.

First Name:	Last Name:	Age:	Class Name:	Grade Average:	Address:
Billy	Davis	17	Leaving Cert 1	70.50	12 High Street, Dublin
Anna	Smith	18	Leaving Cert 1	80.00	19 Lower Street, Dublin
Georgina	Moriarty	17	Leaving Cert 1	90.00	5 Middle Street, Dublin

Call the toString() method on each object in turn.

Using static (class) variables and methods, answer the following questions:

- How many student objects have been created?
- What is the average student grade for the class?

Please turn over

```
First Name: Billy
Last Name: Davis
Age: 17
Class Name: Leaving Cert 1
Grade Average: 70.5
Address: 12 High Street, Dublin

First Name: Anna
Last Name: Smith
Age: 18
Class Name: Leaving Cert 1
Grade Average: 80.0
Address: 19 Lower Street, Dublin

First Name: Georgina
Last Name: Moriarty
Age: 17
Class Name: Leaving Cert 1
Grade Average: 90.0
Address: 5 Middle Street, Dublin

There are: 3 students.
The grade average is: 80.166664
```

Store your source code in a folder named **JFT10Ex8**

Question 9

Create a Java class to model a **Smartphone**. The application will have two classes:

- Smartphone
- Test

The **SmartPhone** class will have the following attributes. The access modifier is public.

Attribute	Data Type
name	String
manufacturer	String
price	float
releaseDate	String
creditRemaining	float
isFullyCharged	boolean

Create the following instance methods in the **SmartPhone** class.

Please turn over



Method Signature	Description	Return Type	Parameter(s)
public void makeCall(String noToCall)	Simulate the making of a phone call. Print the following message to the console, "Dialling number: ". Append the method parameter (noToCall) to the output string.	void	(String noToCall)
public void topUpCredit(float topUpAmount)	<p>Simulate the topping up of credit on a smartphone.</p> <p>Add the topUpAmount to the current value of the instance variable credit_Remaining.</p> <p>Print the message, "You have successfully topped up by: ?".</p> <p>"Your new balance is: ?".</p>	void	(float topUpAmount)
public void charge()	<p>Simulate the charging of a Smartphone.</p> <p>Set the value of the instance variable isFullyCharged to true.</p> <p>Print the message, "The smartphone is now fully re-charged."</p>	void	None
public String toString()	Return a String capturing the values of the	String	None



	instance variables of a String object.		
--	---	--	--

In the second class of the application (**Test**), create the main method.

Create three objects from the **SmartPhone** class with the following attributes.

Object 1:

Name	Data Type
name	Nooks 610
manufacturer	Nooks
price	100.00
releaseDate	"04/06/2008"
creditRemaining	0.00
isFullyCharged	false

Object 2:

Name	Data Type
name	H3C Sensation 610
manufacturer	H3C
price	110.00
releaseDate	"14/07/2012"
creditRemaining	0.00
isFullyCharged	false

Object 3:

Name	Data Type
name	Simsung Desire
manufacturer	Simsung
price	190.00
releaseDate	"14/07/2015"
creditRemaining	0.00
isFullyCharged	false

- Top up the credit on smartphone 1 by €15.00 and recharge the battery.
- Call the toString() method on each object reference.
- Using static variables and methods, determine the number of Smartphone objects created and the average price of a Smartphone.

Store your source code in a folder named **JFT10Ex9**



```
You have successfully topped up by: 15.0
Your new balance is: 15.0
The smartphone is now fully re-charged.

Name: Nooks 610
Manufacturer: Nooks
Price: 100.0
Release Date: 04/06/2008
Credit Remaining: 15.0
Charge Status: true

Name: H3C Sensation 610
Manufacturer: H3C
Price: 110.0
Release Date: 14/07/2012
Credit Remaining: 0.0
Charge Status: false

Name: Sinsung Desire
Manufacturer: Sinsung
Price: 190.0
Release Date: 14/07/2015
Credit Remaining: 0.0
Charge Status: false

3 smartphones have been created.
The average price of a smartphone is: 133.33333
```

Question 10

Create a class named **IQ**, to capture the state of an individual's IQ score.

Instance variables should be created to store the following pieces of information.

Instance Variable	Data Type	Purpose	Scope
name	String	Stores the individual's name.	private
age	int	Stores the individual's age.	private
iQScore	float	Stores the individual's IQ score.	private

Accessor Methods:

- Setter and Getter methods must be declared to allow external classes to access the values held in private instance variables.
- Data validation should be added for the age and IQ score instance variables.
- It can be assumed that a valid age is between (1-115).
- A valid IQ score can be assumed to be between (1-200).
- A default value of zero for both age and IQ score must be set in case of incorrect data values.

Please turn over



Other Methods:

Create a method named, `IQReport()`, to return a message based on an individual's IQ score. If an individual's score is less than one hundred, their score is said to be below average. If an individual scores between one hundred and one hundred and fifty, they are said to have an above average IQ score. An IQ score above 150 is said to be exceptional.

Create a `toString()` method to capture the state of an object created from the `IQ` class.

Test Class:

Create a separate class named **Test** and include the main method.

Create three objects from the **IQ** class:

Object 1

Instance Variable	Value
name	James Devine
age	46
iQScore	87

Object 2

Instance Variable	Value
name	Helena Smith
age	126
iQScore	101

Object 3

Instance Variable	Value
name	Daniel Jones
age	27
iQScore	1510

For each object reference, call the `toString()` method and `IQReport()` method in turn.

Store your source code in a folder named **JFT10Ex10**

```
JFT10Ex10.java: 1000
Name: James Devine Age: 46 IQ Score: 87.0
Your score of 87.0 is below average.
Name: Helena Smith Age: 0 IQ Score: 101.0
Your score of 101.0 is above average.
Name: Daniel Jones Age: 27 IQ Score: 0.0
Your score of 0.0 is below average.
```



Question 11

A television is an object with a state (current channel, current volume, power on/off etc.) and behaviours (change channel, adjust volume, turn on/off etc.)

Create a class in Java (**Television.java**) to model a television set with the following attributes and behaviours.

Attribute	Data Type	Note
channel	int	default channel is 1
volumeLevel	int	default volume level is 1
on_off	boolean	default value is false

Behaviour	Method Name	Note
Set Channel No	setChannel(int newChannel)	Set a new channel, only if the TV is turned on and the channel number to be set is between 1 and 120.
Get Channel No	getChannel()	Return the current channel no.
Set Volume Level	setVolume(int newVolumeLevel)	Set a new volume level, only if the TV is turned on and the new volume level to be set is between 1 and 7.
Get Volume Level	getVolume()	Return the current volume level.
Turn set on / off	setOn_Off(boolean on_off)	If a boolean value of true is passed, the on_off boolean instance variable should be set to true, otherwise, it should be set to false.
Get state of on_off value	getOn_Off()	Return the value of the on_off instance variable value.
Move channel up by 1	channelUp()	Move the channel up by 1 setting, but only if the TV is turned on and the current channel number is < 120.
Move channel down by one.	channelDown()	Move the channel down by one setting, but only if the TV is turned on and the current channel setting is > 1.
Increase volume by one unit.	volumeUp()	Increase the volume by one unit, but only if the TV is turned on and the current volume level is < 7.



Decrease volume by one unit.	volumeDown()	Decrease the volume by one unit, but only if the TV is turned on and the current volume level is > 1.
------------------------------	--------------	---

- Include a toString() method to capture the state of an object created from the class.
- Create a second class named **Test.java**.
- Create a Television object with the following attributes.

Attribute	Value
channel	100
volumeLevel	5
on_off	true

- Call the following methods using the object reference:

channelUp();
channelDown();
volumeUp();
volumeUp();
volumeDown()
setOn_Off(false) // turn off tv
toString()

```
Channel = 100, VolumeLevel = 6, on = false
```

Store your source code in a folder named **JFT10Ex11**

End of Exercises