IBDP Computer Science

Assessment of Learning

Introduction to Object Oriented Programming

OOP 3

Candidate Name

Level (please circle) HL SL

Instructions

Do not use a calculator.

Attempt every question.

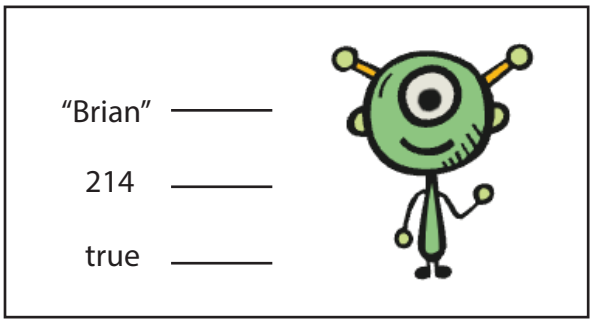
Cross out work that you do not wish the examiner to consider.

Raise your hand if you have a question.

If you finish early, take a break and then look over your answers again.

Points available : SL - 35
HL - 40

Alien objects have 3 attributes: name, age, friendly indicator.
These attributes are hidden inside Alien objects and initialised via parameters in the constructor method.
The only way for other classes to interact with attribute data is via accessor and mutator methods.
Only the age can change after an Alien is constructed.



1 Design the entire Alien class.

[4]

2 Outline the term "data hiding".

[2]

Aliens live on Planets. Here is the Planet class:

```
public class Planet{

	private int population;
	private String[] cities;
	private boolean polluted;

	public Planet( ){
		/*code not shown*/
	}

	public int getPopulation(){
		/*code not shown*/
	}

	public String[] getCities(){
		/*code not shown*/
	}

	public String getCity(int index){
		if(index >= 0 && index < cities.length){
			return cities[index];
		}else{
			return null;
		}
	}

	public void setPolluted(boolean polluted){
		this.polluted = polluted;
	}

	public void setPolluted(){
		this.polluted = true;
	}

	public boolean getPolluted(){
		return polluted;
	}
}
```

3 i Using the information available in the `Planet` class, draw the corresponding UML diagram.

[3]

ii There is an example of polymorphism in the `Planet` class. Name and describe this form of polymorphism.

[2]

4 Write a line of code to construct an instance of the `Planet` class.

[3]

5 The `Planet` class may be termed an abstraction of a real planet. Explain why this is so.

[2]

6

A Planet, `p1`, has the following cities:

```
{"xaloh", "zetagh", "aloosh", "bzevra", "kalalala", "mebrakala"}
```

A Planet class method `getCityIndex` implements a binary search algorithm to search the cities array for a city. If it finds the city in the array, it returns the index of the city, otherwise it returns -1.

For example, using the above array of city names, a call to:
`p1.getCityIndex("boola")` will return -1.

whereas a call to `p1.getCityIndex("aloosh")` will return 2.

i. Explain why the `getCityIndex` method would not work as intended on the above array of Strings, even if the method had been implemented correctly using the binary search algorithm.

[1]

ii. Implement the `getCityIndex` method of the `Planet` class.

[5]

7 Another `Planet` class method, `sortCities`, sorts the cities array into alphabetical order and returns a reference to it.

Design the `sortCities` method. Do not use any Java library routines to sort the array.

To compare Strings, use Java's compareTo method. For example if we have 2 Strings, s1 and s2 then s1.campareTo(s2) will return a negative number if s1 is before s2 in the alphabet, a positive number if s1 is after s2 in the alphabet and 0 if s1 and s2 are equal.

If you are not sure how to do this and decide to use relational operators such as > and <, you can still achieve points, just not full points.

[5]

8 The `Planet` class is being modified. It will now need to also store the size of a Planet. Describe the changes that will need to be made to the Planet class to accomodate this modification.

A new private attribute will need to be added to the class.
The constructor method will have to be modified to initialise the new attribute.
Mutator and constructor methods will need to be designed to allow other classes access to the attribute.

[3]

9

A collection of Planet objects, `planets`, exists.
Construct an algorithm that will calculate and output a count of how many Planets in planets are polluted.
Assume that the Planet class has a new method, `getPolluted()` which returns a boolean value.
pre-condition: the collection has at least on Planet

[5]

10

HL Extension

An Array List of Planets, `planets`, exists.

It is sent to a method, `removePolluted()` as a parameter.
The `removePolluted` method will remove all Planets from the ArrayList which are polluted.
It will return a reference to the resulting ArrayList.
Implement the `removePolluted` function.
pre-condition: there may be zero or any number of Planets in `planets`.

[5]