

*Solutions and Marking Scheme for
Self-test: Knowing how much I know about Java*

Answer TWO questions.

Full marks will be awarded for complete answers to TWO questions.

There are 50 marks available on this paper

Electronic calculators may be used. The make and model should be specified on the script and the calculator must not be programmed prior to the examination.

NOTE: Other correct solutions are possible and acceptable.

Question 1 (a) The name clash of x does not give a problem. [1/3]

Because the x s are both formal parameters. Java treats the x of methodOne as different from the x of methodTwo. [2/3]

(b) Abstraction enables a class or method to concentrate on the essentials of its behaviour and interface to the world. The details can be filled in at a later stage. [2/4]

Inheritance enables a new class to be defined based on an existed class. Also, when a new class is defined, an option is left open to define additional versions of it later. Finally, since the new version of an old class will inherit the properties of original class, the new class can be smaller and neater. [2/4]

(c) The set of the statements is inefficient. Every statement has to be executed although subject can have only one of the values listed. Once subject has a value, checking for the others can be skipped. [3/7]

A better alternative is to use switch-case or if-else. [4/7]

```
switch (subject) {  
    case 'M': System.out.println("Mathematics"); break;  
    case 'E': System.out.println("English"); break;  
    case 'P': System.out.println("Physics"); break;  
    case 'C': System.out.println("Chemistry"); break;  
}
```

or

```
if (subject == 'M') System.out.println("Mathematics");  
else if (subject == 'E') System.out.println("English");  
else if (subject == 'P') System.out.println("Physics");  
else if (subject == 'C') System.out.println("Chemistry");
```

(d) The default value of ch is not E for the second segment of codes (while). [4]

(e) `boolean Between10n50 (int [] A, int n) {`
 `boolean OK = true;`
 `for (int i=0; i<n; i++) {`
 `if (A[i] < 10) then {`
 `OK = false;`
 `break;`
 `}`
 `else if (A[i] > 50) then {`
 `OK = false;`

```
        break;
    }
}
return OK;
}
```

[7]

Question 2 (a)

Red
Green + Red

[2+3/5]

- (b) *Subclass* is the class that has been extended from another class. It is a *child* class in a hierarchical structure.

For example, class B is a subclass of A. [1+1/6]

Superclass is the class that has been extended to form another class. It is a *parent* class in a hierarchical structure.

For example, class A is the superclass of B. [1+1/6]

A subclass can supply its own version of a method already in a superclass. When this happens, the method is said to be *override*.

For example, method Red() in class B is an *override*. [1+1/6]

- (c) [2]

```
double mark [] = new double [1000];
```

- (d) [5]

```
void display (double [] marks) {  
    for (int i=0; i<1000; i++) {  
        System.out.println (marks[i]);  
    }  
}
```

- (e) An exception is an object that signals that some unusual condition has occurred. [3]

The four essential things are:

(1) Try: Create a block around statements where their execution may cause an exception and preface them with the keyword try. [1/7]

(2) Catch: Follow the try statement with one or more handlers prefaced by the keyword catch. [1/7]

(3) Throw: If it is not dealt with at all, the exception will be automatically passed up to the calling method. If it is caught and dealt with partially, the exception can still be passed up with a throw statement. [1/7]

(4) Declare: Mention in the method declaration which exceptions it may throw back to its caller. [1/7]

Question 3 (a) Error: "=" should be "==". [2/4]

Correction: [2/4]

```
if number == 3 System.out.println("full!")
else System.out.println("Try again!");
```

(b) Error: The cases can only be actual values, not conditions. [2/4]

Correction: [2/4]

```
if (number > 0) System.out.println("Positive");
else if (number == 0) System.out.println("Zero");
else System.out.println("Negative");
```

(c) Error: A typed method must have a return statement. [2/4]

Correction: [2/4]

```
class C {
    int getIncreased (int i) {
        A[i] = ++ A[i];
        return A[i];
    }
    private int A[] = new int [5];
}
```

(d) Error: Since signs is NOT smaller than 0 in the first place, no '=' is printed. [2/4]

Correction: [2/4]

```
for (int signs=5; signs>0; signs--) {
    System.out.print('=');
}
System.out.println();
```

Note: Other correct solutions are possible. For example:

```
for (int signs=0; signs<5; signs++) {
    System.out.print('=');
}
System.out.println();
```

(e) Error: M.length returns 100, but M[100] does not exist. [2/4]

Correction: [2/4]

```
int M [] = new int [100];
for (int i=0; i<M.length; i++) {
    M[i] = M[i] * 0.3;
}
```

(f) Error: (a) Hello is a class so there should not be () after it.

[1/5]

(b) Package javagently.* needs to be imported in order to use

```
BufferedReader in = Text.open(System.in);
Text.prompt("What is your name?");}
```

[2/5]

Correction:

[2/5]

```
import java.io.*;
import javagently.*;
```

```
Class Hello {
    public void main (String [] args) throws IOException {
        BufferedReader in = Text.open(System.in);
        Text.prompt("What is your name?");
        String name = Text.readString(in);
        System.out.println("Hello " + name);
    }
}
```