UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2003

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C123

PROGRAMMING II

Friday 9 May 2003, 16:00
Duration: 90 minutes
(Reading time 5 minutes)

Answer THREE questions

Paper contains 4 questions
Calculators not required

Section A *(Use a separate answer book for this Section)*

1a i) Define the Abstract Data Type (ADT) *Stack,* and the ADT *Queue*, and explain the difference between them.

ii) Give the *Stack*'s access procedure headers with post-conditions, specifying also the exception cases.

iii) Give the access procedure headers with post conditions, specifying also the exception cases, for the ADT *Queue.*

iv) Give the axioms that any implementation of the ADT *Queue* must satisfy.

b Assume the availability of ADTs *Stack* and *Queue* with their access procedures. Use these access procedures to write the Java code for the following high-level methods:

i)
```
Queue stackToQueue(Stack s)
    //post: returns a queue containing the items of s, so that the order of
    //       retrieval of the items from the returned Queue is the same as the
    //       retrieval order from s.
```

ii)
```
Queue mergeQueues( Queue q1, Queue q2)
    //post: returns a Queue containing all the items of q1 and q2 in order:
    //       first q1 item, first q2 item, second q1 item, second q2 item...
    //       followed by the remaining items of the longer of q1 and q2 if
    //       q1 and q2 have different lengths.
```

The two parts carry, respectively, 60%, 40% of the marks.

2 The Abstract Data Type *FamilyTree* is to be implemented in Java. Details stored about each Person are firstname, surname, date of birth and date of death (the latter may have no value).

The interface is as follows:

```
public interface FamilyTreeInterface {

    public printFamily();
    //pre: the FamilyTree is not empty
    //post: prints name and date of birth of all
    //individuals in the FamilyTree

    public addSpouse(Person p, Person s);
    //post:  adds s to FamilyTree as spouse of p.

    public addChild(Person p, Person c);
    //post: adds c to FamilyTree as child of p.

    }  //end of FamilyTreeInterface
```

a Give declarations for all data structures used in your implementation.

b Implement the FamilyTreeInterface in the `FamilyTree` class. Write Java code for the access procedures given above, and also the code for the two further methods:

a constructor method `FamilyTree` which takes a Person `ancestor` and returns a new FamilyTree which contains just `ancestor`.

an auxiliary method `findPerson` which takes a name `pname` and returns a reference to the entry with `pname` in the FamilyTree.

c Draw a diagram showing how your implementation would structure the data for the following family:

Joe Smith, date of birth 1.1.1938, married to Emily Jones, d.o.b. 2.2.1940. Joe and Emily's children are Peter, d.o.b. 10.5.1963, Rachel d.o.b. 10.6.1964 and Rebecca d.o.b. 31.10.1967

Rachel is married to Michael Brown, d.o.b.10.7.1962.
Rachel and Michael's children are Joanne d.o.b. 10.7.1992 and Jake d.o.b.2.8.1995, date of death 15.10.2002

Rebecca is married to Robert Wong d.o.b. 2.3.1961

The three parts carry, respectively, 30%, 60%, and 10% of the marks.

Section B *(Use a separate answer book for this Section)*

3a For each of the following Java keywords describe their purpose and illustrate, through a *short* segment of code, how they might be used.

i) static

ii) instanceof

iii) new

iv) extends

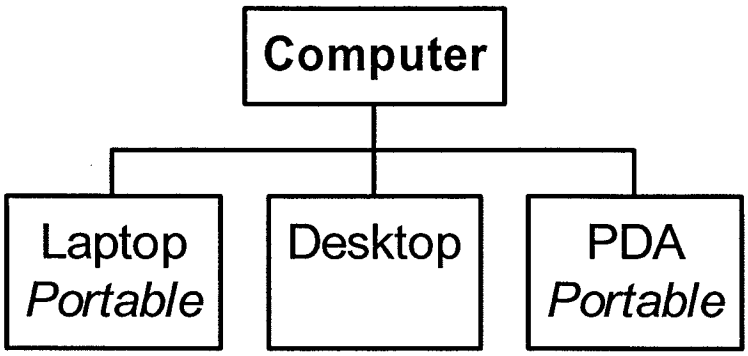
v) throw

b Java provides several mechanisms for data hiding (or encapsulation) within a class and between super- and sub- classes through *visibility* modifiers. Describe each of these modifiers and the effect each has on its associated method or variable declaration.

c Describe the difference between *overriding* and *overloading* a method. Illustrate your answer with reference to methods for setting the state of a `Date` class and the calculation of the area of a `Circle` and a `Doughnut` (a circular disk with a central circular hole). Include relevant Java code in your answer.

The three parts carry, respectively, 50%, 25%, 25% of the marks.

4 The following diagram illustrates the relationship between a set of classes.



In the diagram the top name in each box indicates the name of the class. A name in bold indicates that it is an abstract class and a name in normal type that it is a complete concrete class. If an additional name in italics appears in the box then it is that of an interface that has to be implemented by the class. Lines within the diagram indicate inheritance, i.e. `Computer` is a super-class to `Desktop`.

The interface `Portable` is defined by:

```
interface Portable {
  double mass();
}
```

All classes must have at least one constructor, containing at least one String argument that is used to set the name variable within each `Computer` class instance. Write Java code to produce the following:

a A `Computer` class containing a method to set and access a `String` variable called `name`, and an abstract method:

```
boolean hasKeyboard();
```

b A `Laptop` class which implements the `Portable` interface and has a keyboard, and a mass provided through the constructor.

continued ...

c A `main` method to instantiate an array of type `Computer[]` from a file with the following format:

```
laptop titan 1.9
desktop neptune
pda polaris 0.15
```

where the portable items are annotated with their mass in kg. Ask the user to enter the filename through the keyboard and then read the specified file from disk. Handle exceptions relating to:

- any missing files by asking the user to re-enter the filename
- any incorrectly formatted lines by reporting the problem, where it occurred and exiting from the program

Other exceptions should be passed out of the program. You may assume the existence of a `PDA` class.

d A method to return the number of portable computers, i.e. objects that are an instance of a class which is a sub-class of the `Computer` class that implement the `Portable` interface.

```
static int portableComputers(Computer [] computers)
{


}
```

The four parts carry, respectively, 20%, 20%, 40% and 20% of the marks.