UNIVERSITY OF WESTMINSTER

IMPORTANT: The exam paper is provided in two versions according to the Course you are enrolled on.
Answer only the questions in the version corresponding to the programming language that you have been taught during the module:

1) Java Version (pages 2 - 4): BSc Computer Science, BSc Multimedia Computing, BSc Digital Media Development or BEng Software Engineering

2) C++ Version (pages 5 - 7): BSc Computer Games Development

DO NOT TURN OVER THIS PAGE
UNTIL THE INVIGILATOR INSTRUCTS YOU TO DO SO

<u>Section 1</u>

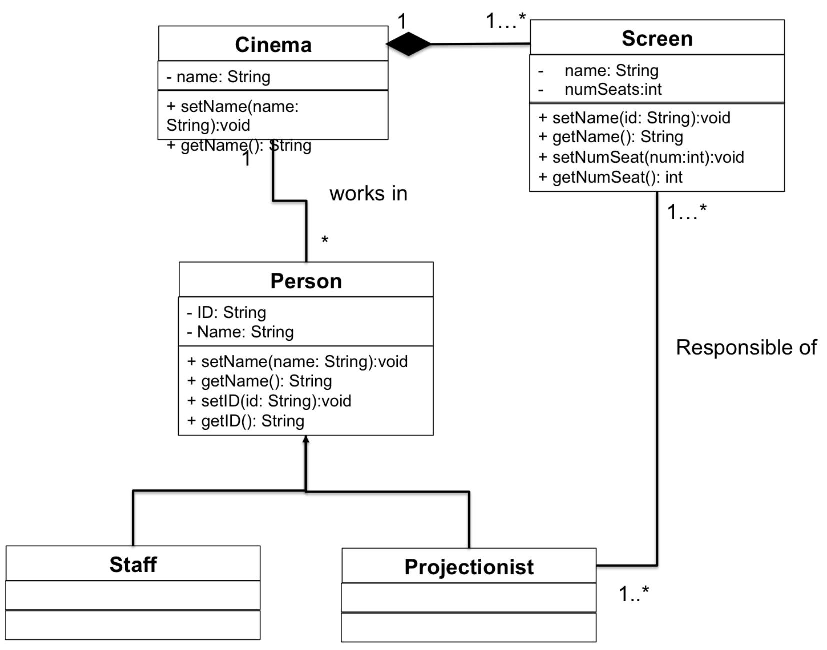
Answer all the following questions in this section if you are enrolled on **BSc Computer Science, BSc Multimedia Computing or BEng Software Engineering**

1) Define a UML class diagram, of the problem presented below:

A cinema has a name and consists of different screens. Each screen has a name and a number of seats. If a cinema stops to exist, and as a result, it needs to be erased, then the screens should also be erased.

A person working in the cinema has a name and a unique ID and can be a member of staff or a projectionist. Each projectionist is responsible for some screens.

(12 points)



2) Briefly describe the access modifiers and how they are represented in the UML diagram.

a. # Protected: the feature is accessible to the class itself, all the classes in the same package, and all its subclasses.
b. – Private: the feature is only accessible within the class itself.
c. + Public: the feature is accessible to any class

(tot 9 points)

3) Consider the following class definition:

```
public class CinemaScreen {
      public String name;
      public int seatsNum; // from 1 to 100
};
```

a. Redesigned CinemaScreen class using more appropriate access modifiers according the encapsulation principle.

```
public class CinemaScreen {
private String name;
private int seatsNum; // from 1 to 100
};
```

(5 points)

b. Provide a constructor for the CinemaScreen class that will initialise the instance variables to suitable (valid) start values.

```
public CinemaScreen (){
   this.name = “Default Name”;
   this.seatsNum = 1;
}
```

(5 points)

c. Provide signatures for set and get methods for the instance variables.

```
public void setName(String name);
public String getName();
public void setSeatsNum(int seatsNum);
public int getSeatsNum();
```

(5 points)

d. Write bodies for the methods for which you provided signatures in question (c), above. Note that for the set method, the implementation must prevent the instance variable from being set to invalid values.

```
public void setSeatsNum(int seatsNum){
   if(seat >= 1 && seat<=100)
         this.seatsNum = seatsNum;
   else
         System.out.println (“invalid number of seats”);
}

public int getSeatsNum(){
   return seatsNum;
}

public void setName(String name){
   this.name = name;

public String getName(){
   return name;
}
```

(8 points)

e. Write a main method that instantiates the radio class and use the set and get methods you designed

```
public class MainClass{
public static void main(String[] args){
   ScreeCinema screen = new ScreenCinema();
   screen.setName ("Screen 1");
   screen.setSeatsNum(50);
System.out.println("The screen name is " + screen.getName()+ "the number of
seats is " + screen.getSeatsNum() );
}}
```

(5 points)

4) Suppose that `class Cat` and `class Dog` are subclasses of `class Animal`. Which of the following are legal?

```
a. Dog d = new Animal();

b. Animal a = new Dog ();

c. Dog d = new Cat();

b
```

(5 points)

5) Consider the following class named Book and reply the following questions.

```
public class Book {

    private String name;
    private String author;
    private int pageNumber;

    public Book(String name){
        this.name = name;
    }

    public void setAuthor(String author){
        this.author = author;
    }

    public void setPage(int pageNumber){
        this.pageNumber = pageNumber;
    }

    public String getAuthor(){
        return author;
    }

    public String getName(){
        return name;
    }


    public int getPage(){
        return pageNumber;
    }

}
```

a. Explain how you could compare different books in Java according to the number of pages and provide an implementation for that.

```
public class Book implements Comparable<Book>{
    private String name;

...

    public int compareTo(Book book){
        return(this.pageNumber – book. pageNumber);
    }


}
```

b. Provide a main class where more than one book are instantiated and stored in an array (or a list) of Books. For each book, set the fields with some values. Print on the screen the ordered list according to the number of pages.

```
public class Main{
public static void main(String[] args){
book b1 = new Book(“OOP”);
b1.setPage(40);
book b2 = new Book(“Database”);
b2.setPage(30);
book b3 = new Book(“Algorithm”);
b3.setPage(25);

Book[] list = new Book[3];
list[0] = b1;
list[1] = b2;
list[2] = b3;

Arrays.sort(list);
System.out.println("Sorting of Book list:\n"+Arrays.toString(list));

}
}
```

6) Explain two methods on how you can terminate a thread in Java. You can provide also examples in java code.

1) Add a boolean variable which indicates whether the thread should continue or not
Provide a set method for that variable which can be invoked by another thread

```
public class Test extends Thread
{
      private volatile boolean exit = false;

        public void run() {
            while(!exit)
                System.out.println("Thread is running.....");
            }
            System.out.println("Thread is stopped....");
        }

        public void stopThread(){
            exit = true;
        }
             }

public class TestThread {

    public static void main(String[] args) throws InterruptedException {

        Test myThread = new Test();
        myThread.start();
        try{
        Thread.sleep(100);
        }
        catch(InterruptedException e){
            System.out.println("Interrupted ");
        }

      myThread.stopThread ();


    }
             }
```

2) We can use an interrupt() method to notify a thread that it should terminate
This method set a Boolean variable in the thread data structure
In the run method we should check for interruptions for each iteration:

```
public void run(){
        for (int i = 0; i<=Repetition && !Thread.interrupted(); i++)
        {
                // Do work
        }
        // Clean up
}
```

This is an example:

```
public class Test extends Thread
{

        public void run() {
            while(!Thread.interrupted()){
                System.out.println("Thread is running.....");
            }
            System.out.println("Thread is stopped....");
        }

}

public class TestThread {

    public static void main(String[] args) throws InterruptedException {
        Test myThread = new Test();
        myThread.start();
        try{
        Thread.sleep(100);
        }
        catch(InterruptedException e){
            System.out.println("Interrupted ");
        }

      myThread.interrupt();

    }
}
```

(10 points)

7) Explain briefly what is encapsulation in object oriented programming.

- wrapping the data (variables) and code acting on the data (methods) together as a single unit.
- The variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

(6 points)

8) What are the differences between error, exception and runtime in java?

Three Throwable subclass categories are possible:

– Error (extends Throwable): Serious error that is not usually recoverable.

– Exceptions (extends Throwable): Error that must be caught and recovered from.

– Runtime Exceptions (extends Exception) : Error that may be caught if desired.

(6 points)

Other question:

1) Explain the concept of abstract in Java and describe abstract methods and classes.

2) What is the constructor in a class?

3) Explain the method wait(), notify(), notifyAll() used in multithreading

4) Explain the concept of Polymorphism with also some examples.

5) Write an example code where you instantiate a JFrame, make it visible and set the size.

6) For the class in exercise 5 sort the list in alphabetic order according to the name.

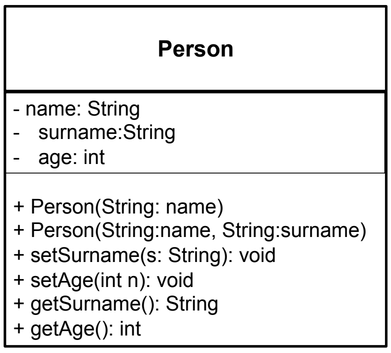
7) Define a UML class diagram of the problem presented below:

Each customer has a name and address and can make an order and the corresponding payment. The payment can be of three kinds: cash, check or credit. Each order contains the order details, such as the date that has been done and the status and also its associated items that have been purchased.

8) How synchronized work in java? Provide an example to show why it can be used and how.

9) Wait and notify: Explain in java why they are used and provide an example.

10) Write the java code for the following class represented in UML notation:



```
Person
-------------------------------------------
- name: String
-   surname:String
-   age: int
-------------------------------------------
+ Person(String: name)
+ Person(String:name, String:surname)
+ setSurname(s: String): void
+ setAge(int n): void
+ getSurname(): String
+ getAge(): int
```

11) Explain the inheritance principle and bring some examples.

12) Describe the event handling model and how you will implement in Java.

13) Describe how to start the Thread in Java. Provide also an example in Java code.

Section 2

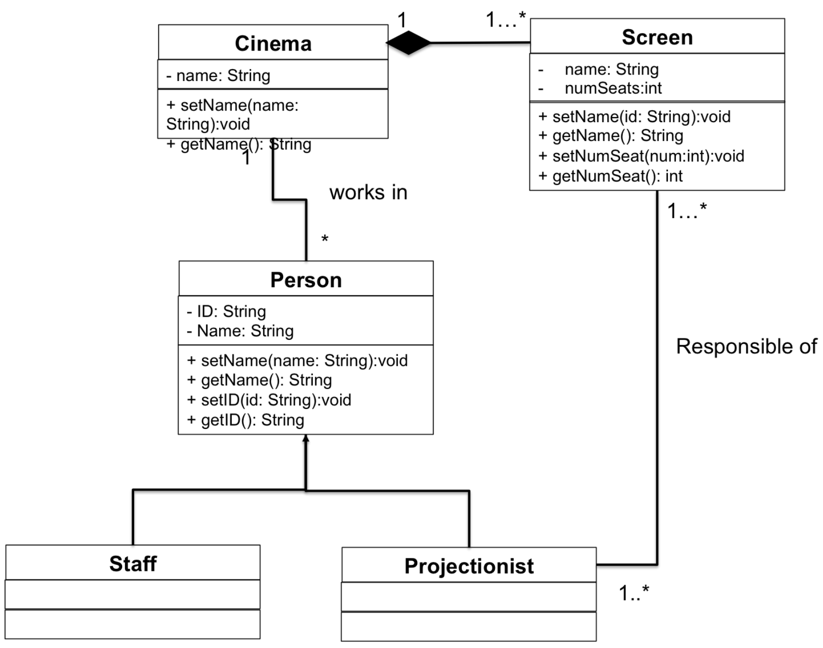
Answer all the following questions in this section if you are enrolled on **BSc Computer Game Development**

1) Define a UML class diagram, of the problem presented below:

A cinema has a name and consists of different screens. Each screen has a name and a number of seats. If a cinema stops to exist, and as a result, it needs to be erased, then the screens should also be erased.

A person working in the cinema has a name and a unique ID and can be a member of staff or a projectionist. Each projectionist is responsible for some screens.

(12 points)



2) Briefly describe the access modifiers and how they are represented in the UML diagram.

a. # Protected: the feature is accessible to the class itself, all the classes in the same package, and all its subclasses.
b. – Private: the feature is only accessible within the class itself.
c. + Public: the feature is accessible to any class

(9 points)

3) Consider the following class definition:

```
public class CinemaScreen {
      public String name;
      public int seatsNum; // from 1 to 100
};
```

4) For all parts of this question, consider a class representing a CinemaScreen. It should have two data fields:

- name of type string;
- seatsNumber of type int, with only values from 0 to 100 valid.

a. Provide a CinemaScreen class that uses appropriate access modifiers according the encapsulation principle.

```
class CinemaScreen {
private:
 string name;
 int seatsNum; // from 0 to 100
};
```

(5 points)

b. Provide a constructor for the CinemaScreen class that will initialise the instance variables to suitable (valid) start values.

```
public CinemaScreen (){
    this.name = "";
    this.seatsNum = 0;
}
```

(5 points)

c. Provide signatures for set and get methods for the instance variables.

```
public:
 void setName(string name);
 string getName();
void setSeatsNum(int seatsNum);
 int getSeatsNum();
```

(5 points)

d. Write bodies for the methods for which you provided signatures in question (b), above.

Note that for the set method, the implementation must prevent the instance variable from being set to invalid values.

```
void setSeatsNum(int seatsNum){
   if(seat >= 1 && seat<=100)
          this.seatsNum = seatsNum;
   else
          cout << "invalid number of seats" << endl;
}
```

```
                int getSeatsNum(){
                   return seatsNum;
                }

                void setName(String name){
                   this.name = name;

                string getName(){
                   return name;
                }
```

(8 points)

e. Write a main method that instantiates the CinemaScreen class and use the set and get methods you designed

```
                int main(){
                   ScreeCinema* screen = new ScreenCinema();
                   Screen->setName (“Screen 1”);
                   Screen->setSeatsNum(50);
                cout<< “The screen name is “ << screen.getName()<< “the number of seats
                is “ << screen->getSeatsNum() );

}}
```

(5 points)

5) Suppose that `class Dog` and `class Cat` are subclasses of `class Animal` and that we have the following objects:

```
Animal animal;
Dog dog;
Cat cat;
```

Which of the following are legal?

```
    a. Dog* d = &animal;

    b. Animal* a = &dog;

    c. Cat* c = &dog;
```

```
b
```

(5 points)

5) Consider the following class named Book representing a football team in a league.

```
class Book {
    public:
```

```
    String name;
    int pageNumber;

  public:
    Book(string name, int pageNumber){
      this.name = name;
      this. pageNumber = pageNumber;
  };

    void setPage(int pageNumber){
      this. pageNumber = pageNumber;
  };

  String getName(){
      return name;
  };
  int getPage(){
      return pageNumber;
  };
};
```

a. Implement the `operator<` for Book class in order to provide a way to compare different Book by their points in the league first, then by name.

```
bool operator() (const Book &x, const Book &y){
return (x. pageNumber < y. pageNumber) ||
((x. pageNumber == y. pageNumber) && x.name < y.name));
};
```

(10 points)

b. Write a main method where some Books are instantiated and stored in an array (or list) of Book. For each Team, choose the name and number of pages and order the list.

```
#include <list>
#include <string>
#include <algorithm> // for sort()


int main(){
std::list<Book> listofBook = { Book("OOP", 100),
                           Book ("Database", 109),
                           Book ("algorithm", "78");}

// Sort List by default criteria i.e < operator of Book
listofTeam.sort();

std::cout<<"****After Sorting By page number ****"<<std::endl;
   for(Book & team : listofBook)
         std::cout<<team.name<< " :: "<<team.pageNumber<<std::endl;


}
```

(6 points)

3 Teams are created: 3 points
Three teams are added into Array or a list: 3 points

6) Explain what is a mutex and the basic mutex operation in C++

(Lecture week 09)

(10 points)

7) Explain briefly what is encapsulation in object oriented programming.

- wrapping the data (variables) and code acting on the data (methods) together as a single unit.
- The variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

(6 points)

8) Explain the use of try catch in C++ and provide an example.

(lecture week 07)

(6 points)

Other questions:

1) Explain the concept of virtual in c++ and provide some examples.

2) What is the constructor in a class?

3) Explain how to create a thread in C++

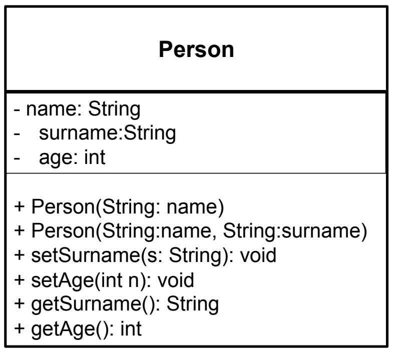
4) Explain the concept of Polymorphism with also some examples.

5) For the class in exercise 5 sort the list in alphabetic order according to the name.

6) Define a UML class diagram of the problem presented below:

Each customer has a name and address and can make an order and the corresponding payment. The payment can be of three kinds: cash, check or credit. Each order contains the order details, such as the date that has been done and the status and also its associated items that have been purchased.

7) Write the java code for the following class represented in UML notation:



8) explain the inheritance principle and bring some examples.