1. Implement the Java classes (with full details of fields and constructors) which are required to simulate the following problem:

1. Implement a `VotingMachine` class that can be used for a simple election. It should have methods to clear the machine state, to vote for a Labour, to vote for a Conservative, and to get the number of votes for each of the two parties.
2. Assuming that your machine is biased towards one of the two parties, modify the `vote` method(s) to add a random number to the number of votes for that party.

[12 marks]

Answer

This is an exercise from your tutorial sessions!

2. Suppose you use a class Clock with private instance variables hours and minutes. How can you access these variables in your program from another class? Select all of the correct answers from below.

1. You access these variables as `Clock.hours` and `Clock.minutes`.
2. You access these variables as `Clock.getHours()` and `Clock.getMinutes()`.
3. If `clock1` is an instance of the `Clock` class, you access these variables as `clock1.hours` and `clock1.minutes`.
4. You cannot access these variables directly.

[8 marks]

Answer

The last option is the only correct answer because these are private members of the class. The first one is incorrect, the fields are not static therefore they cannot be accessed without the creation of the object. The same applies for the second option. The third option is incorrect because another class cannot access the private members of a different class.

3. Consider the following classes:

```
public class Player {
    private String name;
    private double salary;
    ...
    public Player(String theName, double theSalary){ ... }
    public String getName(){ ... }
    public double getSalary(){ ... }
    public void display(){ ... }
}

public class BaseballPlayer extends Player {
    private int atBats;
    private int hits;
    private int homeRuns;
    ...
    public BaseballPlayer(String theName, double theSalary,
                 int theAtBats, int theHits, int theHR) {
       // --> *** MISSING STATEMENT GOES HERE *** <--
       atBats = theAtBats;
       hits = theHits;
       homeruns = theHR;
    }
    public double getBattingAvg() { ... }
    public void display() { ... }
}
```

Which of the following statement(s) can be used to complete the constructor in the `BaseballPlayer` class? Justify your answers of why each one of the statements below can be used or not.

1. `super(theName, theSalary);`
2. `super();`
3. `super(name, salary);`
4. `super(theName, theSalary, theAtBats, theHits, theHR);`

[5 marks]

Answer

The first one is the only correct answer. The second option is incorrect because the parent class does not define a constructor without arguments. The third option is incorrect because there are `name` and `salary` variables which in scope. The last option is incorrect because there are is constructor with 5 arguments defined in the parent class.

4. Consider the following abstract class:

```
public abstract class Card {
    private String name;

    public Card() {
        name = "";
    }

    public Card(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public abstract boolean isExpired();


    public String format() {
        return "Card holder: " + name;
    }
}
```

a. Use this class as a superclass to implement a hierarchy of related concrete (i.e. non-abstract) classes:

- `CreditCard` with fields `int pinNumber`, `int number`.
- `DriverLicense` with field `int expirationYear`.
- `Passport` with fields `String birthLocation`, `int expirationYear`.

Write definitions for each of the subclasses. For each subclass, the instance variables should be declared private. Leave the bodies of the constructors blank for now.

Make the assumption that a credit card does not expire.

[9 marks]

b. Implement constructors for each of the three subclasses. Each constructor should call the superclass constructor to set the name.

[9 marks]

c. Implement a method `main` which creates an `ArrayList` with the name `creditCardList` which can hold a number of `CreditCard` objects as elements. Populate the list with 5 credit card objects.

[5 marks]

d. Expand the functionality of class `CreditCard` so that the following code (without any changes) will display on the console ALL the details (`pinNumber`, `number` and `name`) of all the credit card objects in the list `creditCardList` stored in the previous subquestion:

```
for (CreditCard c: creditCardList)
    System.out.println(c);
```

[8 marks]

Answer

This is an exercise from your tutorial sessions! For the last subquestion, you need to define the `toString()` in the class. There were similar tutorial exercises for this part too.

5. Briefly explain the meaning of the `final` Java keyword when it is used for fields, methods and classes. What is the difference between `final` fields, methods and classes and their non-`final` correspondents?

[8 marks]

Answer

This is based on the lecture notes! Make sure you study all of the lecture and tutorial material of the module!

6. **a.** Guess what will happen when you attempt to compile and run the following code? (without actually compiling/running it)

```
public class Background extends Thread {
    public static void main(String argv[]) {
        Background b = new Background();
        b.run();
    }

    public void start() {
        for (int i = 0; i < 10; i++) {
            System.out.println("Value of i = " + i);
        }
    }
}
```

1. A compile time error indicating that no run method is defined for the Thread class.
2. A run time error indicating that no run method is defined for the Thread class.
3. The code compiles and at run time the values 0 to 9 are printed out.
4. The code compiles but there is no output at runtime.

[8 marks]

Answer

This is an exercise from your tutorial sessions!

b. Explain what is wrong with the following code and why this is happening. Modify (fix) the code so that it runs as expected.

```
class Thread1 extends Thread {
    public void run() {
        while (true) {
            System.out.println("Hey I am thread: " + getName());

            try {
                Thread.sleep(1000);   // sleep 1 sec
            }
            catch (InterruptedException ex) {
                System.out.println("Thread 1 was interrupted");
            }
        }
    }
}

class Thread2 extends Thread {
    public void run() {
        while (true) {
            System.out.println("Hey I am thread: " + getName());

            try {
                Thread.sleep(1000);   // sleep 1 sec
            }
            catch (InterruptedException ex) {
                System.out.println("Thread 2 was interrupted");
            }
        }
    }
}

class ProblematicThreads {
    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        Thread2 t2 = new Thread2();

        t1.run();
        t2.run();
    }
}
```

[8 marks]

Answer

This is an exercise from your tutorial sessions!

7. Write a Java program to solve the following problem:

A palindromic number reads the same both ways. For example number 90109 reads the same whether you start from right to left or left to right. The largest palindrome made from the product of two 2-digit numbers is $9009 = 91 \times 99$.

Find the largest palindrome made from the product of two 3-digit numbers.

[20 marks]

Answer

Just a sample solution, this could be done in many different ways!

```
class Palindrome {
    // returns true only if the given argument is a palindrome
    public boolean isPalindrome(int number) {
        /* convert int to a string for easy access to individual digits
           alternatively one could extract the digits from the int
           diretly using division by 10 and calculating remainders
         */
        String s = "" + number;
        for (int i=0; i <= s.length()/2; i++) {
            int len = s.length();
            if (s.charAt(i) != s.charAt(len-i-1))
                return false;  // nope, it is not a palindrome
        }

        // it is a palindrome!
        return true;
    }
}

class PalindromeQuestion {
    public static void main(String[] args) {
        Palindrome pal = new Palindrome();

        int max_product = 0;
        /* produce all 3-digit products and check
           if each one is a palindrome */
        for (int i = 100; i < 1000; i++)
            for (int j = 100; j < 1000; j++) {
                int product = i*j;
                if (pal.isPalindrome(product))
                    if (product > max_product)
                        max_product = product;
            }

        System.out.println(max_product);
    }
}
```

END OF MOCK PAPER