Question 1. [17 MARKS]

Below is a partially completed class modelling cars. Read it carefully. For this question, you will write code calling methods of the class, and also write additional methods for the class.

```
public class Car {
    private String maker;            // Chrysler, Ford, Toyota, etc.

    private double kmPerLitre;      // the fuel efficiency: how far
                                    // it can go on one litre of gas

    private double litresOfGas;     // gas currently in the tank
    private int tankSize;           // the maximum the tank can hold

    public Car(String m, double k, int t) {
        maker = m;
        kmPerLitre = k;
        tankSize = t;
    }
} // class Car
```

Part (a) [3 MARKS]

In a *single* java statement: construct an object representing a Toyota that has a 50 litre gas tank and gets 15.5km per litre, and declare a reference `c1` to this object.

SOLUTION:

```
Car c1 = new Car("Toyota", 15.5, 50);
```

MARKING SCHEME:

- You got marks for having `Car c1 = new Car(...)` (you lost marks for missing `Car`, *i.e.*, having just `c1 = ...` or for missing `c1`).
- You got marks for having 3 parameters (you didn't get this one if you had missing `""`, or extra `""` or units (L,km/L)).
- You got marks for having 3 parameters in the correct order.

COMMENTS: (common mistakes)

- `Toyota` not in quotes
- `Car c1 = new Car(String Toyota, double 15.5, int 50)` (argument types are *not* part of a method call)

Part (b) [3 MARKS]

Write a method `fillErUp()` for the `Car` class, which sets the amount of fuel currently in the tank to the maximum amount.

SOLUTION:

```
public void fillErUp() {
    litresOfGas = tankSize;
}
```

MARKING SCHEME:

- You lost marks for missing or having incorrect or additional return type.
- You lost marks for saying `static`.
- You lost marks for having parameters.
- You lost marks for having a completely wrong body, or for smaller errors in the body (such as having "`litresOfGas = double(tankSize)`" or "`litresOfGas = 50`").

COMMENTS: (common mistakes)

- Putting `litresOfGas` as a parameter.
- Missing curly brackets `{}`.
- Using a nonexistent variable (*e.g.*, `amountOfFuel`) instead of `litresOfGas`.
- Having a return type.

Part (c) [3 MARKS]

Write a method `fuelInTank()` that could be called as demonstrated in the line of code

```
double litres = c1.fuelInTank();
```

and that does what its name suggests.
SOLUTION:

```
public double fuelInTank() {
    return litresOfGas;
}
```

MARKING SCHEME:

- You lost marks for missing or having incorrect or additional return type.
- You lost marks for having parameters.
- You lost marks for omitting `return` statement.
- You lost marks for returning the wrong amount.

COMMENTS: (common mistakes)

-
  ```
  public double fuelInTank(double x) {
      return x;
  }
  ```

-
  ```
  public double fuelInTank(double x) {
      x = litresOfGas;
      return x;
  }
  ```
- Printing the value using `System.out.println` instead of returning it.

The next 2 parts are about a method `drive()`. The parameter for this method indicates the number of kilometers the user hopes to drive. If there is enough gas for the trip, the method removes that much gas and returns `true`. If there is not enough gas, the method uses up all the gas and returns `false`.
Here is an example of a call to the method. In this example, `c1` is a `Car` reference to an existing `Car` with some gas.

```
if (c1.drive(140.2)) {
    System.out.println("We made it to London!");
} else {
    System.out.println("We ran out of gas!");
}
```

Part (d) [2 MARKS]

Write the prototype for `drive()`.
SOLUTION:

```
public boolean drive(double kms)
```

MARKING SCHEME:

- You got marks for having `boolean drive`.
- You got marks for having a parameter of type `double`.

COMMENTS: It was ok to have `boolean drive(double)` and not give the parameter a name. If you had part (d) wrong or missing but you had the correct prototype in part (e), then you would get some marks for (d). A common mistake was to write `Boolean` instead of `boolean`.

Part (e) [6 MARKS]

Write the body of `drive()`.
SOLUTION:

```
double gasNeeded = kms / kmPerLitre;
if (litresOfGas >= gasNeeded) {
    litresOfGas -= gasNeeded;
    return true;
} else {
    litresOfGas = 0;
    return false;
}
```

MARKING SCHEME:

- You got marks for having the correct condition and `if` syntax.
- You got marks for adjusting `litresOfGas` in the true clause.
- You got marks for adjusting `litresOfGas` in the false clause.
- You got marks for returning `true` or `false` in each clause.
- You got marks for doing this return correctly *after* doing the other stuff.

Comments: (common mistakes)

- using the parameter as if it is the litres needed instead of the distance
- saying only "`return (kms/kmPerLitre >= litresOfGas)`"

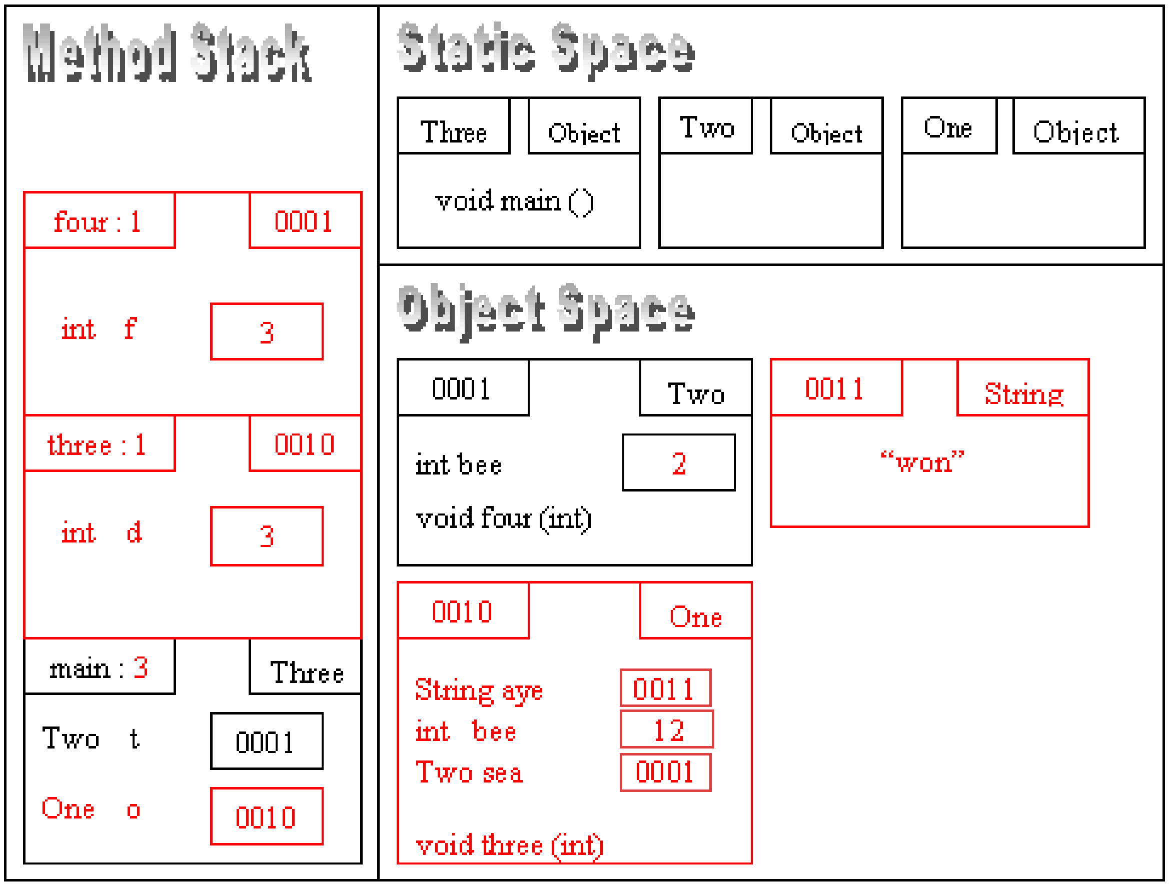
-
  ```
  litresOfGas -= kms/kmPerLitre;
  return kms/kmPerLitre >= litresOfGas;
  ```
- having `return` statements before the assignments to `LitresOfGas`
- missing "`=`" in `litresOfGas >= gasNeeded`
- using local variables from other methods
- using undeclared variables
- using `140.2` as the minimum or maximum distance (the method should work correctly in *all* cases, not just for the example)
- having incorrect `return` statements

Question 2. [12 marks]

The following is a Java program and the memory model that results after executing statement 1 in the `main` method. Update the diagram to show the state of memory after statement 1 in method `four` executes for the first time (this is initiated by the call in statement 3 of method `main`.) As in Assignment 2, you may *cheat* by representing `String`s as text inside variable boxes.

```
class One {                                |   class Two {
    private String aye;                    |       private int bee;
    private int bee;                       |       public Two(int d) {
    private Two sea;                       |              bee = d;
    public One(String a, int b, Two c) {   |       }
        aye = a;                           |       public void four(int f) {
        bee = b;                           |           bee -= f;         // stmt 1
        sea = c;                           |           int dee = 5;      // stmt 2
    }                                      |       }
    public void three(int d) {             |   } // class Two
        sea.four(d);        // stmt 1      |
    }                                      |
} // class One                             |   public class Three {
                                           |       public static void main(String[] args) {
                                           |           Two t = new Two(5);              // stmt 1
                                           |           One o = new One("won",12,t);     // stmt 2
                                           |           o.three(3);                      // stmt 3
                                           |       }
                                           |   } // class Three
```

SOLUTION:



MARKING SCHEME:

- Method Stack [6 marks]

  main:
  - marks for line number `3` (common mistake)
  - marks for `One o`
  - marks for address `0010`—this can be any address, but it must match the address used in the `One` object (it does not need to be binary, but it must be a unique address)

  three:
  - marks for `three:1`
  - marks for using the address of Object reference `o` in `main` (common mistake)
  - marks for `int d`
  - marks for `d` containing value `3`

  four:
  - marks for `four:1`
  - marks for using the address of Object reference `t` in `main` (common mistake)
  - marks for `int f`
  - marks for `f` containing value `3`

- Object Space [6 marks]

  Object Two: marks for `bee` containing value `2`

Object One: – marks for class `One` in top right hand corner

- marks for `String aye`
- marks for `int bee`
- marks for `Two sea`
- marks for `void three (int)` (common error: missing parameter type `int`)
- marks for either putting `won` into the box next to `String aye`, or correctly making a new String object at a new address containing `won`
- marks for `12` for `bee` (common error: value `9`)
- marks for `0001` for `sea` (same address as `t` in `main`) (common error: use `t` instead of its address)

Comments: (common mistakes)

- duplicating prototypes (including the same method more than once in a single object)
- forgetting quotes around `String` values, if you were "cheating"
- having an additional object from class `Two` in the Object Space
- missing a box on the Method Stack, or crossing out a box that should be there
- having 2 or more boxes for `main`
- forgetting the type of variables in the Object Space
- Generally, people did quite poorly for the Method Stack.

Question 3. [14 marks]

Here is a Java program, which compiles and runs without error messages. Write the output from the program in the space below it.

```
public class Q3 {
    public static void main(String[] args) {
        String s = "Alan Turing";
        String t = s;
        System.out.println(s.substring(1, 3));
        s = s.toUpperCase() + " " + t.toLowerCase();
        System.out.println(t);
        mister(s);
        System.out.println(s);
        Person p1 = new Person("Alan Turing", "Student");
        Person p2 = new Person("Alan Turing", "Computer Scientist");
        Person p3 = p1;
        p3.changeOccupation("Secret Agent");
        p1.printOccupation();
        p2.printOccupation();
        p3.printOccupation();
    }
    public static void mister(String s) {
        s = "Mr. " + s;
        System.out.println(s);
    }
}
```

```
public class Person {
    private String name;
    private String occupation;
    public Person(String n, String o) {
        name = n;
        occupation = o;
    }
    public void changeOccupation(String o) {
        occupation = o;
    }
    public void printOccupation() {
        System.out.println(occupation);
    }
}
```

Solution:

```
la
Alan Turing
Mr. ALAN TURING alan turing
ALAN TURING alan turing
Secret Agent
Computer Scientist
Secret Agent
```

Marking Scheme: You got marks for each correct line of output. Output that was close but not exact would usually result in *no* mark for that line because it generally meant that you didn't know what caused the output. If you were off by one with the `substring` call, you would get part marks. If you thought `s` changed due to the `substring` but were consistent in using this afterwards, you were only penalized once.

Comments: (common mistakes)

- misunderstanding `substring` (missing some letters, having some letters extra)
- missing `System.out.println` in method call `mister(s)`
- missing `changeOccupation` to "`Secret Agent`" for `p1` (since `p1 == p3`)
- "`Alan Turing, Secret Agent`"
- having "`Mr. Alan Turing`" twice
- missing capitals for the first "`Alan Turing`"
- reversing the order of some calls
- having "`ALAN TURING alan turing`" as second line
- having "`Mr. ALAN TURING alan turing`" last
- forgetting capital letters on second and third lines (`ALAN TURING`)
- using quotation marks, or having extra characters