Aims

The assignment will exercise:

- The use of the basic language features of Java
- Designing and building a solution to a specification

Background

As part of a larger project, building an online hotel management system, you need to design and implement a system for storing and calculating some key information about a hotel.

Task

Your task is to create a prototype which allows capturing knowledge about the current state of a hotel.

You will need at least three classes:

`Hotel`
: This should store all the essential information about a hotel, including a name and some rooms.

`Room`
: This should store the number of beds in a room.

`Bed`
: This should store the size of a bed (i.e. single or double).

You will need to instantiate multiple objects of these classes. Pay attention to the design principles that you have been taught during the module. For instance, every `Hotel` should have a `Name` property, which will be accessible by `get/set` methods and be stored in a `private` member variable.

For later stages of this coursework, you will build a simple user interface. This is a prototype system, so the user interface does **NOT** need to be complex; a simple command-line interface using `Scanner` and `println` statements is entirely sufficient. However, as your classes may be used in several different situations, none of `Hotel`, `Room` nor `Bed` should require any user interaction; use of `println` or `Scanner` within these three classes will be considered an error, and will be penalised.

There are a number of features that you should attempt to achieve; please complete these in the order given.

HotelReport

Create a class called `HotelReport` which, when given a `Hotel` object will produce a short, textual report describing the name of the hotel, the number of rooms and, for each room, lists the number and size of the beds.

HotelTest

The initial version of your system should allow you to create objects describing one hotel, with several rooms, each with one or more beds. The completed `Hotel` object should be able to calculate its maximum occupancy. You should write a simple class called `HotelTest` which contains a `main` method which demonstrates this working.

HotelConfigure

Create a simple class called `HotelConfigure`. This should have a `main` method which provides a simple user interface, which allows you to choose a number of rooms, and for each room choose the number and size of beds.

After the user has completed this task, the `HotelConfigure` class should print display this information back to the user.

Vacancy

Add a property called `hasVacancies` to `Hotel`. A hotel has a vacancy if any room is vacant. Make any changes you need in the other classes to support this property, including `HotelReport`, `HotelTest` and `HotelConfigure`

SUBMISSION NOTES:

Your submission should consist of a set of Java source code files. No other files are necessary, nor will be accepted. NESS has been set up to accept a relatively large number of Java files; this is simply an upper limit and not an expectation.

Where class names are given in the specification, you should use these class names exactly as given. These named classes should all be `public` and, therefore, in files with the same names.

All classes should be in the same Java package, which may be the default package.

MARKING SCHEME:

Marks will be allocated as follows:

- Core Data Model: 70%
- Report/Test/Configure classes: 30%

For each of these:

- Correctness and Completeness: 70%
- Style and Readability: 30%

You will attract marks for:

- Good use of get/set methods
- Appropriate use of types
- Correct calculations
- Good use of object orientation
- Good error checking
- Use of privacy
- Appropriate constructors
- Correct indentation
- Good variable naming
- Clear use of comments

Please note that, that we can also mark negatively where the code is directly against the specification (inappropriate use of user interaction as described above, for example), or against the design principles described in the lectures (for example, overuse of the `static` keyword).