High-Quality Code Exam – Computers

You are given a computer building system. It can create different types of computers from various manufacturers. Every computer is built from components. Current supported components in the system are:

- **RAM memory** – has maximum **amount**. The RAM memory can **save** and **load** a single integer value
- **CPU** – has **number of cores**. The CPU can generate random number between given minimum and maximum value and save it to the RAM. The CPU can calculate the square number for a given number loaded from the RAM. There are currently two types of CPUs in the system – **32 bit** and **64 bit**. The **32 bit** one can calculate the power of two for numbers between **0** and **500**, inclusive. The **64 bit** one can calculate the power of two for numbers between **0** and **1000**. In case of lower value, the video card should print "**Number too low.**" In case of higher value, the video card should print "**Number too high.**" **The CPU should get and save data from the RAM. The result of square number operation should be outputted to the video card.**
- **Hard drive** – has **capacity**. The hard drive can **save text data on given integer address**. The hard drive can **load data from given integer address**. Currently there is one type of hard disk in the system – HDD. There is also a raid array, which works as a collection of same hard drives. Its capacity is equal to one of the hard drives capacity. When saving text data to an address, the raid saves it on all hard drives in its collection (for backup reasons). When loading text data from address, the raid returns the data from the first working hard drive in its collection. In case of trying to load data and there are no hard drives in the collection, the raid should throw exception message "**No hard drive in the RAID array!**"
- **Video card** – can **draw text data**. Currently there are two types of video card in the system – **monochrome** and **colorful**. The **monochrome** video cards prints text on the console in **grey**. The **colorful** video cards prints text on the console in **green**.
- **Battery** – can **charge** itself and has information about **percentage power left**. Currently there is only one type of battery in the system – laptop battery. It is created with 50% power left initially. It can charge itself with certain amount but never above **100** and below **0**.
- **Motherboard** – It can load values from the RAM, save values to the RAM and draw on the video card.

Currently the system can build three types of computers:

- **Personal computer** – can use any CPU, RAM and Video card. Can use multiple hard disks. The personal computer can play video games. Currently one video **game** is installed – the computer generates random number between 1 and 10, inclusive. The user enters a number between 1 and 10 and if the numbers are the same as the random number generated, the video card should print "**You win!**" Otherwise, it should print "**You didn't guess the number {0}.**" (where "{0}" is the random number generated).

- **Laptop** – can use any CPU, RAM and Video card. Can use multiple hard disks. Must have a battery. The laptop can charge its battery by a given percentage and every time it charges, the video card should print "**Battery status: {0}%**"
- **Server** – can use any CPU, RAM. Can use multiple hard drives. Always uses monochrome video card. Can process requests. Currently the server can print the square number of given integer by using its CPU and RAM.

Currently the system has these two manufacturers added:

- **Dell** – it manufactures these computers:
  - **PC** – 64 bit CPU with 4 cores, 8GB RAM, one 1000GB hard drive and colorful video card.
  - **Server** – 64 bit CPU with 8 cores, 64GB RAM, one Raid with two 2000GB hard drives.
  - **Laptop** – 32 bit CPU with 4 cores, 8GB RAM, one 1000GB hard drive, colorful video card and laptop battery.
- **HP** – it manufactures these computers:
  - **PC** – 32 bit CPU with 2 cores, 2GB RAM, one 500GB hard drive and colorful video card.
  - **Server** – 32 bit CPU with 4 cores, 32GB RAM, one Raid with two 1000GB hard drives.
  - **Laptop** – 64 bit CPU with 2 cores, 4GB RAM, one 500GB hard drive, colorful video card and laptop battery.

You are assigned to create a program that executes a sequence of commands against computers from certain manufacturer. The first line of the console will contain the manufacturer's name and one PC, one server and one laptop is created from that manufacturer. If invalid manufacturer is passed, an exception with message "**Invalid manufacturer!**" should be thrown. Each next command consists of a single text line executed on these three computers. Each command has only one parameter – a single number separated from the command name with " " (space). If invalid command is passed, an "**Invalid command!**" message should be written on the console. The commands are described below:

- **Charge *[number]*** – charges the laptop by given percentage
- **Process *[number]*** – makes request to the server and calculates the square of a number
- **Play *[number]*** – plays the game on the personal computer with the given number
- **Exit** – indicates the end of the input sequence of commands. "**Exit**" stops the commands processing without any output. It is always the last command in the input sequence.

Input

The input data comes from the console. It consists of manufacturer name and a sequence of commands, each staying at a separate single line. The input ends by the "**Exit**" command.

The **input data will always be valid** and in the described format. There is no need to check it explicitly.

Output

The output should be on the console. It should consist of the outputs from each of the commands from the input sequence.

Constraints

- The command names will always be separated by the space symbol " " from the parameters.
- Numbers will be valid 32-bit signed integer numbers.

Examples

Input example	Output example
HP Play 5 Play 6 Charge 10 Charge 0 Charge -100 Charge 200 Charge -1 Process -1 Process 1001 Process 1000 Process 500 Process 16 Exit	You didn't guess the number 3. You win! Battery status: 60% Battery status: 60% Battery status: 0% Battery status: 100% Battery status: 99% Number too low. Number too high. Number too high. Square of 500 is 250000. Square of 16 is 256.
Dell Play 5 Play 6 Charge 10 Charge 0 Charge -100 Charge 200 Charge -1 Process -1 Process 1001 Process 1000 Process 500 Process 16 Exit	You win! You didn't guess the number 9. Battery status: 60% Battery status: 60% Battery status: 0% Battery status: 100% Battery status: 99% Number too low. Number too high. Square of 1000 is 1000000. Square of 500 is 250000. Square of 16 is 256.

Lenovo Play 5 Play 6 Charge 10 Charge 0 Charge -100 Charge 200 Charge -1 Process -1 Process 2001 Process 2000 Process 1000 Process 500 Process 16 Exit	You didn't guess the number 8. You didn't guess the number 1. Battery status: 60% Battery status: 60% Battery status: 0% Battery status: 100% Battery status: 99% Number too low. Number too high. Square of 2000 is 4000000. Square of 1000 is 1000000. Square of 500 is 250000. Square of 16 is 256.

Problem 1. Code Refactoring

Refactor the source code to improve its quality following the best practices introduced in the course "High-Quality Programming Code". **You are not allowed to change the IMotherboard interface.** Find common refactoring problems, including:

- **Extract core classes and interfaces in a different assembly**
- **Introduce constants**
- **Remove useless constant**
- **Middle man**
- **Divergent change**
- **Better naming of classes, variables, fields, properties and parameters**
- **Fix namespace names**
- **Remove code duplication**
- **Fix compile errors and warnings**

16 points

Problem 2. StyleCop

Fix all **StyleCop** issues you can find**.**

8 points

Problem 3. Design Patterns

Add these six design patterns to the solution and describe how you implement them in the provided text file (*Documentation.txt*):

- **Simple Factory** (2 points)
- **Strategy** (3 points)
- **Template Method** (4 points)
- **Abstract Factory** (4 points)
- **Composite** (4 points)
- **Mediator** (5 points)

22 points

Problem 4. Unit Testing

Design and implement **unit tests for**:

- `LaptopBattery.Charge()`
- `Cpu.SquareNumber()`
- `Cpu.rand()`

Any other code is not required to be tested. The **code coverage** should be at **least 90% for the listed methods.** Be sure to test all major execution scenarios + all interesting border cases and special cases. Use VSTT and VS code coverage.

At least one of your tests should use mocking (JustMock or Moq).

12 points

Problem 5. Code Documentation

Document the `IMotherboard` interface and all methods defined in **the `IMotherboard` interface** using XML documentation. Any other code documentation is not required.

6 points

Problem 6. Performance Bottlenecks

Find **two performance bottlenecks** and briefly describe them in the provided text file (*Documentation.txt*).

Fix the problems if possible. If not, put a comment in the code about the problem.

6 points

Problem 7. Bug Fixing

Debug the code and **find and fix the three bugs** in it. Describe them in the provided text file (*Documentation.txt*).

10 points

Problem 8. New Features

Implement a 128 bit CPU (a brand new technology). Its max available value for the square number operation should be 2000.

Implement creation of Lenovo computers with the following parameters:

- **PC** – 64 bit CPU with 2 cores, 4GB RAM, one 2000GB hard drive and monochrome video card.
- **Server** – 128 bit CPU with 2 cores, 8GB RAM, one Raid with two 500GB hard drives.
- **Laptop** – 64 bit CPU with 2 cores, 16GB RAM, one 1000GB hard drive, colorful video card and laptop battery.

10 points

Problem 9. SOLID

Describe all SOLID principles used in your code in the provided text file (*Documentation.txt*).

10 points

Ensure the program works correctly after all modifications!

Don't forget to attach *Documentation.txt* file among with your refactored source code.

Good luck!