Collections (15 marks):

The API for the `Money` class is given below. Each instance of this class represents an amount of dollars and cents. The amount of cents will be an integer between 0 and 99 (inclusive).

Constructor Summary	
`Money()` Construct a Money amount with 0 dollars and 0 cents.	
`Money(int d, int c)` Construct a Money amount with `d` dollars and `c` cents.	
Method Summary	
`void`	`add(Money amount)` Increase this Money amount by `amount`.
`boolean`	`isGreaterThan(Money amount)` Returns `true` if this Money amount is greater than `amount`.
`void`	`subtract(Money amount)` Decrease this Money amount by `amount`.

The API for the `Product` class is given below. Each instance of a Product has a cost, a Stock Keeping Unit (SKU), and a Description.

Constructor Summary	
`Product(Money cost, String description, int sku)` Construct a new Product with the given `cost`, `description`, and `sku`.	
Method Summary	
`Money`	`getCost()` Returns the cost of this Product.
`String`	`getDescription()` Returns the description of this Product.
`int`	`getSKU()` Returns the Stock Keeping Unit of this Product.

The API for the `Warehouse` class is given below. Each instance of this class contains the Products and the quantity of each Product that are stored in the Warehouse.

Field Summary	
`static int`	`MAX_PRODUCTS` The maximum number of different Products that a Warehouse can store.
Constructor Summary	
`Warehouse ()` Constructs a new Warehouse.	
Method Summary	
`void`	`addProduct (Product product)` Adds the `product` (with 0 units) to the Warehouse inventory. The added product will be numbered as the numProducts Product (numProducts will then be increased by 1).
`void`	`addUnit (int i)` Adds one unit of the i^{th} product to the Warehouse inventory. Products are 0-indexed from 0 to numProducts-1.
`void`	`addUnit (Product product)` Adds one unit of the given `product` to the Warehouse inventory.
`int`	`getNumProducts ()` Returns the number of different Products currently stored in the Warehouse. Products are 0-indexed from 0 to numProducts-1.
`Product`	`getProduct (int i)` Returns the i^{th} product in the Warehouse. Products are 0-indexed from 0 to numProducts-1.
`int`	`getQuantity (int i)` Returns the quantity of the i^{th} product that is currently stored in the Warehouse. Products are 0-indexed from 0 to numProducts-1.

Surname:_____ First name:_____ Student #: _____

Write a code fragment in JAVA that will determine the total value of all Products stored in the given Warehouse.

```
// Money totalValue
// Warehouse warehouse
```

Collections (15 marks):

The API for the `File` class is given below. Each instance of this class represents a file (e.g. a music file) that has a name and a size (in bytes).

Constructor Summary	
`File()` Constructs a new File with no name and zero size.	
`File(String name, int size)` Constructs a new File with the given `name` and the given `size`.	
Method Summary	
`String`	`getName()` Returns the name of this File.
`int`	`getSize()` Returns the size (in bytes) of this File.
`boolean`	`isCompressed()` Returns `true` if this File is compressed, `false` otherwise.

The API for the `Compressor` class is given below. The methods allow Files to be compressed and decompressed. Note: to use a File (e.g. to play a music file), it must be decompressed, but it may be stored in a compressed state to save space.

Method Summary	
`static File`	`compress(File file)` If the given `file` is uncompressed (i.e. normal size), returns a compressed copy of the `file`. Otherwise, returns `null`.
`static File`	`decompress(File file)` If the given `file` is compressed, returns a decompressed (i.e. normal size) copy of the `file`. Otherwise, returns `null`.

The API for the `MemoryCard` class is given below. Each instance of this class can store up to 100 Files that can take up to 100 Mbytes of space. Note: files are 0-indexed.

Field Summary	
`static int`	`MAX_FILES` The maximum number of Files that can be stored on a MemoryCard.
`static int`	`MAX_SPACE` The maximum total space available for Files to be stored on a MemoryCard.
Constructor Summary	
`MemoryCard()` Constructs a new MemoryCard with zero Files.	
`MemoryCard(File[] files)` Attempts to construct a new MemoryCard with the given `files`. Will add Files in sequence until `MAX_FILES` or `MAX_SPACE` has been reached.	
Method Summary	
`void`	`delete(int index)` Deletes the File at the given `index`.
`File`	`findFile(int index)` Returns the File stored on this MemoryCard at the given `index`. Returns `null` if there is no File at the given `index`.
`int`	`getAvailableMemory()` Returns the amount of memory still available on this MemoryCard.
`int`	`getNumFiles()` Returns the number of Files currently stored on this MemoryCard.
`int`	`getUsedMemory()` Returns the amount of memory currently used by Files on this MemoryCard.
`boolean`	`insertFile(int index, File file)` Attempts to insert the given `file` at the given `index`. Returns `true` if the file is inserted successfully and `false` otherwise (e.g. if there is insufficient space on the MemoryCard). Note: this method will overwrite/delete any existing Files at the given `index`.

Surname:_____ First name:_____ Student #: _____

Write a code fragment in JAVA that will determine the amount of space saved by using compression on the MemoryCard (i.e. for all compressed files, sum their size difference between been compressed and uncompressed).

```
// MemoryCard card;
```