**Java**

```java
class FileSharing {

    public FileSharing(int m) {

    }

    public int join(List<Integer> ownedChunks) {

    }

    public void leave(int userID) {

    }

    public List<Integer> request(int userID, int chunkID) {

    }
}

/**
 * Your FileSharing object will be instantiated and called as such:
 * FileSharing obj = new FileSharing(m);
 * int param_1 = obj.join(ownedChunks);
 * obj.leave(userID);
 * List<Integer> param_3 = obj.request(userID,chunkID);
 */
--------------------------------------------------------------------
```

**JavaScript**

```javascript
/**
 * @param {number} m
 */
var FileSharing = function(m) {

};

/**
 * @param {number[]} ownedChunks
 * @return {number}
 */
FileSharing.prototype.join = function(ownedChunks) {

};

/**
 * @param {number} userID
 * @return {void}
 */
FileSharing.prototype.leave = function(userID) {

};

/**
 * @param {number} userID
 * @param {number} chunkID
 * @return {number[]}
 */
FileSharing.prototype.request = function(userID, chunkID) {

};
```

```
/**
 * Your FileSharing object will be instantiated and called as such:
 * var obj = new FileSharing(m)
 * var param_1 = obj.join(ownedChunks)
 * obj.leave(userID)
 * var param_3 = obj.request(userID,chunkID)
 */
```

--------------------------------------------------------------------

**TypeScript**

```
class FileSharing {
    constructor(m: number) {

    }

    join(ownedChunks: number[]): number {

    }

    leave(userID: number): void {

    }

    request(userID: number, chunkID: number): number[] {

    }
}

/**
 * Your FileSharing object will be instantiated and called as such:
 * var obj = new FileSharing(m)
 * var param_1 = obj.join(ownedChunks)
 * obj.leave(userID)
 * var param_3 = obj.request(userID,chunkID)
 */
```
--------------------------------------------------------------------

**C++**

```
class FileSharing {
public:
    FileSharing(int m) {

    }

    int join(vector<int> ownedChunks) {

    }

    void leave(int userID) {

    }

    vector<int> request(int userID, int chunkID) {

    }
};

/**
 * Your FileSharing object will be instantiated and called as such:
 * FileSharing* obj = new FileSharing(m);
 * int param_1 = obj->join(ownedChunks);
 * obj->leave(userID);
 * vector<int> param_3 = obj->request(userID,chunkID);
 */
--------------------------------------------------------------------
```

**C#**

```
public class FileSharing {

    public FileSharing(int m) {

    }

    public int Join(IList<int> ownedChunks) {

    }

    public void Leave(int userID) {

    }

    public IList<int> Request(int userID, int chunkID) {

    }
}

/**
 * Your FileSharing object will be instantiated and called as such:
 * FileSharing obj = new FileSharing(m);
 * int param_1 = obj.Join(ownedChunks);
 * obj.Leave(userID);
 * IList<int> param_3 = obj.Request(userID,chunkID);
 */
```

----------------------------------------------------------------------

**Kotlin**

```
class FileSharing(m: Int) {

    fun join(ownedChunks: List<Int>): Int {

    }

    fun leave(userID: Int) {

    }

    fun request(userID: Int, chunkID: Int): List<Int> {

    }

}

/**
 * Your FileSharing object will be instantiated and called as such:
 * var obj = FileSharing(m)
 * var param_1 = obj.join(ownedChunks)
 * obj.leave(userID)
 * var param_3 = obj.request(userID,chunkID)
 */
```
-------------------------------------------------------------------

**Go**

```
type FileSharing struct {
```

```go
}

func Constructor(m int) FileSharing {

}

func (this *FileSharing) Join(ownedChunks []int) int {

}

func (this *FileSharing) Leave(userID int)  {

}

func (this *FileSharing) Request(userID int, chunkID int) []int {

}

/**
 * Your FileSharing object will be instantiated and called as such:
 * obj := Constructor(m);
 * param_1 := obj.Join(ownedChunks);
 * obj.Leave(userID);
 * param_3 := obj.Request(userID,chunkID);
 */
```
-------------------------------------------------------------------