```php
<?php
class ResourcePolicyLine    {
    private $resourceType;
    private $resourceId;
    private $resourceTypeAndId;
    private $sequenceNumber;
    private $accessorType; //Mainly Login Id
    private $accessorId;
    private $accessorTypeAndId;
    private $flags;
    private $comments;
    private $line;
    static $__IS_CLONE_MODE = "__is_clone_mode";
    static $__ALL = 0;
    static $__IS_WRITE_ENABLED = 0; //flags bit 0
    static $__IS_READ_ENABLED = 1; //flags bit 1
    public function isReadOnly()    {
        return ($this->isReadEnabled() && ! $this->isWriteEnabled());
    }
    public function setReadOnly($val)   {
        $val = intval("".$val);
        if ($val == 1)  {
            $this->setReadEnabled("1");
            $this->setWriteEnabled("0");
        } else {
            $this->setReadEnabled("0");
        }
    }
    public function isWriteOnly()   {
        return ($this->isWriteEnabled() && ! $this->isReadEnabled());
    }
    public function setWriteOnly($val)  {
        $val = intval("".$val);
        if ($val == 1)  {
            $this->setWriteEnabled("1");
            $this->setReadEnabled("0");
        } else {
            $this->setWriteEnabled("0");
        }
    }
    public function isReadAndWrite()    {
        return ($this->isReadEnabled() && $this->isWriteEnabled());
    }
    public function setReadAndWrite($val)   {
        $val = intval("".$val);
        if ($val == 1)  {
            $this->setReadEnabled("1");
            $this->setWriteEnabled("1");
        } else {
            $this->setReadEnabled("0");
            $this->setWriteEnabled("0");
        }
    }
    public function isWriteEnabled()    {
        return Object::isControlFlagsSetAt($this->flags,
        ResourcePolicyLine::$__IS_WRITE_ENABLED);
    }
```

```
    public function setWriteEnabled($val)    {
        $val = intval("").$val);
        if ($val == 1)  {
            $this->flags = Object::setControlFlagsAt($this->flags,
            ResourcePolicyLine::$__IS_WRITE_ENABLED);
        } else {
            $this->flags = Object::resetControlFlagsAt($this->flags,
            ResourcePolicyLine::$__IS_WRITE_ENABLED);
        }
    }
    public function isReadEnabled() {
        return Object::isControlFlagsSetAt($this->flags,
        ResourcePolicyLine::$__IS_READ_ENABLED);
    }
    public function setReadEnabled($val)     {
        $val = intval("").$val);
        if ($val == 1)  {
            $this->flags = Object::setControlFlagsAt($this->flags,
            ResourcePolicyLine::$__IS_READ_ENABLED);
        } else {
            $this->flags = Object::resetControlFlagsAt($this->flags,
            ResourcePolicyLine::$__IS_READ_ENABLED);
        }
    }
    public function cloneMe()    {
        $data1 = new ResourcePolicyLine(ResourcePolicyLine::$__IS_CLONE_MODE);
        $dataArray = array();
        $dataArray['resourceType'] = $this->resourceType;
        $dataArray['resourceId'] = $this->resourceId;
        $dataArray['resourceTypeAndId'] = $this->resourceTypeAndId;
        $dataArray['sequenceNumber'] = $this->sequenceNumber;
        $dataArray['accessorType'] = $this->accessorType;
        $dataArray['accessorId'] = $this->accessorId;
        $dataArray['accessorTypeAndId'] = $this->accessorTypeAndId;
        $dataArray['flags'] = $this->flags;
        $dataArray['comments'] = $this->comments;
        $dataArray['line'] = $this->line;
        $data1->assignData($dataArray);
        return $data1;
    }
    public function assignData($dataArray)  {
        if (is_null($dataArray)) Object::shootException("Could not Assign Data");
        if (isset($dataArray['resourceType'])) $this->resourceType =
        $dataArray['resourceType'];
        if (isset($dataArray['resourceId'])) $this->resourceId = $dataArray['resourceId'];
        if (isset($dataArray['resourceTypeAndId'])) $this->resourceTypeAndId =
        $dataArray['resourceTypeAndId'];
        if (isset($dataArray['sequenceNumber'])) $this->sequenceNumber =
        $dataArray['sequenceNumber'];
        if (isset($dataArray['accessorType'])) $this->accessorType =
        $dataArray['accessorType'];
        if (isset($dataArray['accessorId'])) $this->accessorId = $dataArray['accessorId'];
        if (isset($dataArray['accessorTypeAndId'])) $this->accessorTypeAndId =
        $dataArray['accessorTypeAndId'];
        if (isset($dataArray['flags'])) $this->flags = $dataArray['flags'];
        if (isset($dataArray['comments'])) $this->comments = $dataArray['comments'];
        if (isset($dataArray['line'])) $this->line = $dataArray['line'];
```

```
}
public function __construct($line)  {
    if ($line == ResourcePolicyLine::$__IS_CLONE_MODE) return;
    $this->line = $line;
    //Briefcase.4,10,Student.0,2,All Students can Read Only [!comma!]
    $lineArr = explode(",", $line);
    if (sizeof($lineArr) != 5) Object::shootException("The Structure of Policy File is
    Undefined at line : [ $line ]");
    $this->resourceTypeAndId = trim($lineArr[0]);
    $this->sequenceNumber = trim($lineArr[1]);
    $this->accessorTypeAndId = trim($lineArr[2]);
    $this->flags = trim($lineArr[3]);
    $comments = str_replace("!comma!", ",", $lineArr[4]);
    $this->comments = trim($comments);
}
public final static function makeLine($resourceTypeAndId, $sequenceNumber,
$accessorTypeAndId, $flags, $comments)   {
    return
    $resourceTypeAndId.",".$sequenceNumber.",".$accessorTypeAndId.",".$flags.",".str_repla
    ce(",", "!comma!", $comments);
}
public final static function createANewResourcePolicyLine($resourceTypeAndId,
$sequenceNumber, $accessorTypeAndId, $flags, $comments)   {
    return (new ResourcePolicyLine(ResourcePolicyLine::makeLine($resourceTypeAndId,
    $sequenceNumber, $accessorTypeAndId, $flags, $comments)));
}
public function synchronize()   {
    $this->resourceTypeAndId = $this->resourceType.".".$this->resourceId;
    $this->accessorTypeAndId = $this->accessorType.".".$this->accessorId;
    $line = $this->resourceTypeAndId;
    $line .= ",".$this->sequenceNumber;
    $line .= ",".$this->accessorTypeAndId;
    $line .= ",".$this->flags;
    $line .= ",".str_replace(",", "!comma!", $this->comments);
    $this->line = $line;
}
public function getResourceType()   { return $this->resourceType; }
public function setResourceType($resourceType)  { $this->resourceType = $resourceType; }
public function getResourceId() { return $this->resourceId; }
public function setResourceId($resourceId)  { $this->resourceId = $resourceId; }
public function getResourceTypeAndId()  { return $this->resourceTypeAndId; }
public function setResourceTypeAndId($resourceTypeAndId)    { $this->resourceTypeAndId =
$resourceTypeAndId; }
public function getSequenceNumber() { return $this->sequenceNumber; }
public function setSequenceNumber($sequenceNumber)  { $this->sequenceNumber =
$sequenceNumber; }
public function getAccessorType()   { return $this->accessorType; }
public function setAccessorType($accessorType)  { $this->accessorType = $accessorType; }
public function getAccessorId() { return $this->accessorId; }
public function setAccessorId($accessorId)  { $this->accessorId = $accessorId; }
public function getAccessorTypeAndId()  { return $this->accessorTypeAndId; }
public function setAccessorTypeAndId($accessorTypeAndId) { $this->accessorTypeAndId =
$accessorTypeAndId; }
public function getFlags()  { return $this->flags; }
public function setFlags($flags)    { $this->flags = $flags; }
public function getComments()   { return $this->comments; }
public function setComments($comments)  { $this->comments = $comments; }
```

```
    public function getLine()    { return $this->line; }
    public function setLine($line)  { $this->line = $line; }
}
class ResourcePolicyManager {
    private $recourcePolicyLines;
    private $resourceControlIndexArray;
    private $accessorControlIndexArray; //f788 -- proceed from here
    ds[accessorTypeAndId][index] = index
    private $resourcePolicyTrackChanges;
    private $database;
    private $conn;
    private $filename;
    private $checksum; //Especially while saving, you need to make sure checksum is the same
    and no anyone has edited while you were still doing some changes
    static $__IS_CLONE_MODE = "_is_clone_mode";
    static $__IS_MARKED_FOR_DELETION = "_@32767@_";
    public function assignData($dataArray)  {
        if (isset($dataArray['resourcePolicyLines'])) $this->resourcePolicyLines =
        $dataArray['resourcePolicyLines'];
        if (isset($dataArray['resourceControlIndexArray'])) $this->resourceControlIndexArray
        = $dataArray['resourceControlIndexArray'];
        if (isset($dataArray['accessorControlIndexArray'])) $this->accessorControlIndexArray
        = $dataArray['accessorControlIndexArray'];
        if (isset($dataArray['resourcePolicyTrackChanges']))
        $this->resourcePolicyTrackChanges = $dataArray['resourcePolicyTrackChanges'];
        if (isset($dataArray['database'])) $this->database = $dataArray['database'];
        if (isset($dataArray['connectionString'])) $this->conn =
        $dataArray['connectionString'];
        if (isset($dataArray['filename'])) $this->filename = $dataArray['filename'];
        if (isset($dataArray['checksum'])) $this->checksum = $dataArray['checksum'];
    }
    public function
    getResourcePolicyManagerForResourceAndAccessor($collectionOfResourceTypeAndIds,
    $collectionOfAccessorTypeAndIds)    {
        return
        $this->getResourcePolicyManagerForResource($collectionOfResourceTypeAndIds)->getResour
        cePolicyManagerForAccessor($collectionOfAccessorTypeAndIds);
    }
    public function getResourcePolicyManagerForAccessor($collectionOfAccessorTypeAndIds)    {
        //ie Login.7 JobTitle.4 Group.5 Group.3
        $data1 = new ResourcePolicyManager("Ndimangwa",
        ResourcePolicyManager::$__IS_CLONE_MODE, "fADhili Ngoya");
        $resourcePolicyLines = array();
        $resourceControlIndexArray = array();
        $accessorControlIndexArray = array();
        $resourcePolicyTrackChanges = array();
        $listOfAccessorTypeAndIds = $collectionOfAccessorTypeAndIds->getCollection();
        foreach ($listOfAccessorTypeAndIds as $accessorTypeAndId)   {
            if (! isset($this->accessorControlIndexArray[$accessorTypeAndId])) continue;
            $indexArray = $this->accessorControlIndexArray[$accessorTypeAndId];
            foreach ($indexArray as $index) {
                $policyLine1 = $this->resourcePolicyLines[$index];
                if
                ($this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()][$polic
                yLine1->getSequenceNumber()] ==
                ResourcePolicyManager::$__IS_MARKED_FOR_DELETION) continue;
                $datasize = sizeof($resourcePolicyLines);
```

```
                $resourcePolicyLines[$datasize] = $policyLine1->cloneMe();
                if ( !
                isset($resourceControlIndexArray[$policyLine1->getResourceTypeAndId()]))
                $resourceControlIndexArray[$policyLine1->getResourceTypeAndId()] = array();

                $resourceControlIndexArray[$policyLine1->getResourceTypeAndId()][$policyLine1-
                >getSequenceNumber()] = $datasize;
                if (!
                isset($accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()]))
                $accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()] = array();

                $accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()][sizeof($acces
                sorControlIndexArray[$policyLine1->getAccessorTypeAndId()])] = $datasize;
            }
        }
        $dataArray = array();
        $dataArray['resourcePolicyLines'] = $resourcePolicyLines;
        $dataArray['resourceControlIndexArray'] = $resourceControlIndexArray;
        $dataArray['accessorControlIndexArray'] = $accessorControlIndexArray;
        $dataArray['resourcePolicyTrackChanges'] = $resourcePolicyTrackChanges;
        $dataArray['database'] = $this->database;
        $dataArray['connectionString'] = $this->conn;
        $dataArray['filename'] = $this->filename;
        $dataArray['checksum'] = "INVALIDATE"; //can not save
        $data1->assignData($dataArray);
        return $data1;
    }
    public function getResourcePolicyManagerForResource($collectionOfResourceTypeAndIds)     {
        // ie Results.4 ResultsGroup.11 Examination.4 , kindly preserve order
        $data1 = new ResourcePolicyManager("Ndimangwa",
        ResourcePolicyManager::$__IS_CLONE_MODE, "fADhili Ngoya");
        $resourcePolicyLines = array();
        $resourceControlIndexArray = array();
        $accessorControlIndexArray = array();
        $resourcePolicyTrackChanges = array();
        $listOfResourceTypeAndIds = $collectionOfResourceTypeAndIds->getCollection();
        foreach ($listOfResourceTypeAndIds as $resourceTypeAndId)    {
            if (! isset($this->resourceControlIndexArray[$resourceTypeAndId])) continue;
            $sequenceNumberArray = $this->resourceControlIndexArray[$resourceTypeAndId];
            $nextSequenceNumber = 10;
            foreach ($sequenceNumberArray as $olderSequenceNumber => $index)     {
                if ($index == ResourcePolicyManager::$__IS_MARKED_FOR_DELETION) continue;
                //Copy Only those which are okay
                if (! isset($resourceControlIndexArray[$resourceTypeAndId]))
                $resourceControlIndexArray[$resourceTypeAndId] = array();
                $policyLine1 = $this->resourcePolicyLines[$index];
                $policyLine1 = $policyLine1->cloneMe();
                $policyLine1->setSequenceNumber($nextSequenceNumber);
                $policyLine1->synchronize();
                $datasize = sizeof($recourcePolicyLines);
                $resourcePolicyLines[$datasize] = $policyLine1;
                $resourceControlIndexArray[$resourceTypeAndId][$nextSequenceNumber] =
                $datasize;
                if (!
                isset($accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()]))
                $accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()] = array();
```

```
                $accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()][sizeof($acces
                sorControlIndexArray[$policyLine1->getAccessorTypeAndId()])] = $datasize;
                $nextSequenceNumber += 10;
            }
            $resourcePolicyTrackChanges[$resourceTypeAndId] =
            $this->resourcePolicyTrackChanges[$resourceTypeAndId];
        }
        $dataArray = array();
        $dataArray['resourcePolicyLines'] = $resourcePolicyLines;
        $dataArray['resourceControlIndexArray'] = $resourceControlIndexArray;
        $dataArray['accessorControlIndexArray'] = $accessorControlIndexArray;
        $dataArray['resourcePolicyTrackChanges'] = $resourcePolicyTrackChanges;
        $dataArray['database'] = $this->database;
        $dataArray['connectionString'] = $this->conn;
        $dataArray['filename'] = $this->filename;
        $dataArray['checksum'] = "INVALIDATE"; //can not save
        $data1->assignData($dataArray);
        return $data1;
    }
    public function merge($resourcePolicyManager1)  {
        return $this;
    }
    public function cloneMe()   {
        $data1 = new ResourcePolicyManager("Ndimangwa",
        ResourcePolicyManager::$__IS_CLONE_MODE, "Fadhili Ngoya");
        $resourcePolicyLines = array();
        $resourceControlIndexArray = array();
        $accessorControlIndexArray = array();
        $resourcePolicyTrackChanges = array();
        //Now Making a copy of Arrays, have to use a different space
        foreach ($this->resourceControlIndexArray as $resourceTypeAndId =>
        $sequenceNumberArray)    {
            $nextSequenceNumber = 10;
            foreach ($sequenceNumberArray as $olderSequenceNumber => $index)    {
                if ($index == ResourcePolicyManager::$__IS_MARKED_FOR_DELETION) continue;
                //Copy Only those which are okay
                if (! isset($resourceControlIndexArray[$resourceTypeAndId]))
                $resourceControlIndexArray[$resourceTypeAndId] = array();
                $policyLine1 = $this->resourcePolicyLines[$index];
                $policyLine1 = $policyLine1->cloneMe();
                $policyLine1->setSequenceNumber($nextSequenceNumber);
                $policyLine1->synchronize();
                $datasize = sizeof($recourcePolicyLines);
                $resourcePolicyLines[$datasize] = $policyLine1;
                $resourceControlIndexArray[$resourceTypeAndId][$nextSequenceNumber] =
                $datasize;
                if (!
                isset($accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()]))
                $accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()] = array();

                $accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()][sizeof($acces
                sorControlIndexArray[$policyLine1->getAccessorTypeAndId()])] = $datasize;
                $nextSequenceNumber += 10;
            }
            $resourcePolicyTrackChanges[$resourceTypeAndId] =
            $this->resourcePolicyTrackChanges[$resourceTypeAndId];
        }
```

```
        //Finished to make a clean reference of dataStructures
        $dataArray = array();
        $dataArray['resourcePolicyLines'] = $resourcePolicyLines;
        $dataArray['resourceControlIndexArray'] = $resourceControlIndexArray;
        $dataArray['accessorControlIndexArray'] = $accessorControlIndexArray;
        $dataArray['resourcePolicyTrackChanges'] = $resourcePolicyTrackChanges;
        $dataArray['database'] = $this->database;
        $dataArray['connectionString'] = $this->conn;
        $dataArray['filename'] = $this->filename;
        $dataArray['checksum'] = $this->checksum;
        $data1->assignData($dataArray);
        return $data1;
    }
    public function __construct($database, $filename, $conn)    {
        if ($filename == ResourcePolicyManager::$__IS_CLONE_MODE) return;
        //Now check file existence
        if (! file_exists($filename)) Object::shootException("The resource filename does not
        exists");
        $this->filename = $filename;
        $this->database = $database;
        $this->conn = $conn;
        //Data Structures
        $this->resourcePolicyLines = array();
        $this->resourceControlIndexArray = array();
        $this->accessorControlIndexArray = array();
        $this->resourcePolicyTrackChanges = array(); //Default False, no changes
        //Dealing with files , line by line
        $file1 = fopen($filename, "r") or Object::shootException("Could not open file for
        reading");
        $linenumber = 0;
        while (($line = fgets($file1)) !== false)   {
            if (trim($line) == "") continue;
            $linenumber++;
            $policyLine1 = new ResourcePolicyLine($line);
            $index = sizeof($this->resourcePolicyLines);
            $this->resourcePolicyLines[$index] = $policyLine1;
            if (!
            isset($this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()]))
            $this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()] = array();

            $this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()][$policyLine
            1->getSequenceNumber()] = $index;
            if (!
            isset($this->accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()]))
            $this->accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()] = array();

            $this->accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()][sizeof($thi
            s->accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()])] = $index;
            if (!
            isset($this->resourcePolicyTrackChanges[$policyLine1->getResourceTypeAndId()]))
            $this->resourcePolicyTrackChanges[$policyLine1->getResourceTypeAndId()] = false;
        }
        fclose($file1);
    }
    public function save()  {
        //Step 1: You need to clear with the checksum
        //Step 2: You need to create a backup of the original file
```

```
        //Step 3: You need to do a local sort , outer sort based on resource types
        $listOfResourceTypeAndIds = array();
        foreach ($this->resourceControlIndexArray as $resourceTypeAndId => $seqNumBlock)    {
            $listOfResourceTypeAndIds[sizeof($listOfResourceTypeAndIds)] = $resourceTypeAndId;
        }
        //Perform Sorting
        $datasize = sizeof($listOfResourceTypeAndIds);
        for ($i = 0; $i < $datasize - 1; $i++)  {
            $swapped = false;
            for ($j = 0; $j < $datasize - $i - 1; $j++) {
                if ($listOfResourceTypeAndIds[$j] > $listOfResourceTypeAndIds[$j+1])    {
                    $swapped = true;
                    $temp = $listOfResourceTypeAndIds[$j];
                    $listOfResourceTypeAndIds[$j] = $listOfResourceTypeAndIds[$j+1];
                    $listOfResourceTypeAndIds[$j+1] = $temp;
                }
            }
            if (! $swapped) {
                break;
            }
        }
        //You need to save now while , checking track Changes
        $file1 = fopen($this->filename, "w") or Object::shootException("Could not open file
        for writing");
        $lineCount = 0;
        foreach ($listOfResourceTypeAndIds as $resourceTypeAndId)   {
            if ($this->resourcePolicyTrackChanges[$resourceTypeAndId])
            $this->sortSequenceNumber($resourceTypeAndId);
            foreach ($this->resourceControlIndexArray[$resourceTypeAndId] as $sequenceNumber
            => $index) {
                if ($index == ResourcePolicyManager::$__IS_MARKED_FOR_DELETION) continue;
                //Do not save this
                //No need to synchronize, it has been synchronized
                $policyLine1 = $this->resourcePolicyLines[$index];
                $line = $policyLine1->getLine();
                $line = str_replace("\n","", str_replace("\r", "", $line));
                $line .= "\n";
                $lineCount++;
                fwrite($file1, $line) or Object::shootException("Could not write to a file
                [$lineCount]");
            }
        }
        fclose($file1);
    }
    public function sortSequenceNumber($resourceTypeAndId)  {
        if (! isset($this->resourceControlIndexArray[$resourceTypeAndId]))
        Object::shootException("The reference resource type and id were not found");
        //Sequence Number Resolution
        $sequenceNumberResolution = array();
        foreach ($this->resourceControlIndexArray[$resourceTypeAndId] as $sequenceNumber =>
        $index) {
            if ($index == ResourcePolicyManager::$__IS_MARKED_FOR_DELETION) continue;
            //I am dealing with only those not targeted for deletion
            $sequenceNumberResolution[sizeof($sequenceNumberResolution)] = $sequenceNumber;
        }
        $datasize = sizeof($sequenceNumberResolution);
        for ($i=0; $i < ($datasize - 1); $i++)  {
```

```
            $swapped = false;
            for ($j=0; $j < ($datasize - $i - 1); $j++) {
                $policyLine1 =
                $this->resourcePolicyLines[$this->resourceControlIndexArray[$resourceTypeAndId
                ][$sequenceNumberResolution[$j]]];
                $policyLine2 =
                $this->resourcePolicyLines[$this->resourceControlIndexArray[$resourceTypeAndId
                ][$sequenceNumberResolution[$j+1]]];
                if ($policyLine1->getSequenceNumber() > $policyLine2->getSequenceNumber())  {
                    $swapped = true;
                    $temp =
                    $this->resourceControlIndexArray[$resourceTypeAndId][$sequenceNumberResolu
                    tion[$j]];

                    $this->resourceControlIndexArray[$resourceTypeAndId][$sequenceNumberResolu
                    tion[$j]] =
                    $this->resourceControlIndexArray[$resourceTypeAndId][$sequenceNumberResolu
                    tion[$j+1]];

                    $this->resourceControlIndexArray[$resourceTypeAndId][$sequenceNumberResolu
                    tion[$j+1]] = $temp;
                }
            }
            if (! $swapped) {
                //No need to continue, it is already sorted
                break;
            }
        }
        //Now it is wiser to resequence the sequence Number into multiple of 10s and
        synchronize
        $newSequenceNumber = 10;
        foreach ($sequenceNumberResolution as $olderSequenceNumber) {

            $this->resourcePolicyLines[$this->resourceControlIndexArray[$resourceTypeAndId][$o
            lderSequenceNumber]]->setSequenceNumber($newSequenceNumber);

            $this->resourcePolicyLines[$this->resourceControlIndexArray[$resourceTypeAndId][$o
            lderSequenceNumber]]->synchronize();
            $newSequenceNumber += 10; //Increments of 10
        }
        $this->resourcePolicyTrackChanges[$resourceTypeAndId] = false;
    }
    public function addANewResourcePolicyLine($resourceTypeAndId, $sequenceNumber,
    $accessorTypeAndId, $flags, $comments)    {
        $policyLine1 = ResourcePolicyLine::createANewResourcePolicyLine($resourceTypeAndId,
        $sequenceNumber, $accessorTypeAndId, $flags, $comments);
        $index = sizeof($this->resourcePolicyLines);
        $this->resourcePolicyLines[$index] = $policyLine1;
        if (! isset($this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()]))
        $this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()] = array();
        if
        (isset($this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()][$policyL
        ine1->getSequenceNumber()])) Object::shootException("Can not create A New Policy for
        resource, Already Exists");

        $this->resourceControlIndexArray[$policyLine1->getResourceTypeAndId()][$policyLine1->g
        etSequenceNumber()] = $index;
```

```
        //AccessorTypeAndId
        if (! isset($this->accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()]))
        $this->accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()] = array();

        $this->accessorControlIndexArray[$policyLine1->getAccessorTypeAndId()][sizeof($this->a
        ccessorControlIndexArray[$policyLine1->getAccessorTypeAndId()])] = $index;
        $this->resourcePolicyTrackChanges[$policyLine1->getResourceTypeAndId()] = true;
    }
    public function deleteAResourcePolicyLine($resourceTypeAndId, $sequenceNumber)  {
        if ($this->isResourcePolicyLineExists($resourceTypeAndId, $sequenceNumber)) {
            $this->resourceControlIndexArray[$resourceTypeAndId][$sequenceNumber] =
            ResourcePolicyManager::$__IS_MARKED_FOR_DELETION;
            $this->resourcePolicyTrackChanges[$resourceTypeAndId] = true;
        }
    }
    public function isResourcePolicyLineExists($resourceTypeAndId, $sequenceNumber) {
        return (isset($this->resourceControlIndexArray[$resourceTypeAndId]) &&
        isset($this->resourceControlIndexArray[$resourceTypeAndId][$sequenceNumber]) &&
        ($this->resourceControlIndexArray[$resourceTypeAndId][$sequenceNumber] !==
        ResourcePolicyManager::$__IS_MARKED_FOR_DELETION));
    }
    public function getResourcePolicyLine($resourceTypeAndId, $sequenceNumber)  {
        $policyLine1 = null;
        if ($this->isResourcePolicyLineExists($resourceTypeAndId, $sequenceNumber))
        $policyLine1 =
        $this->resourcePolicyLines[$this->resourceControlIndexArray[$resourceTypeAndId][$seque
        nceNumber]];
        return $policyLine1;
    }
    public function getResourcePolicyLines()    { return $this->resourcePolicyLines; }
    public function setResourcePolicyLines($resourcePolicyLines) {
    $this->resourcePolicyLines = $resourcePolicyLines; }
    public function getResourceControlIndexArray()  { return
    $this->resourceControlIndexArray; }
    public function setResourceControlIndexArray($resourceControlIndexArray)    {
    $this->resourceControlIndexArray = $resourceControlIndexArray;}
    public function getDatabase()   { return $this->database; }
    public function setDatabase($database)  { $this->database = $database; }
    public function getConnectionString()   { return $this->conn; }
    public function setConnectionString($conn)  { $this->conn = $conn; }
    public function getFilename()   { return $this->filename; }
    public function setFilename($filename)  { $this->filename = $filename; }
    public function getChecksum()   { return $this->checksum; }
    public function setChecksum($checksum) { $this->checksum = $checksum; }
}
?>
```