Assignment 7

```
import java.util.ArrayList;
import java.util.Scanner;

class PriceException extends RuntimeException{
   PriceException(String s){
      super(s);
   }
}

class EssentialCommodityException extends RuntimeException{
   EssentialCommodityException(String s){
      super(s);
   }
}

class GradeMismatchException extends RuntimeException{
   GradeMismatchException(String s){
      super(s);
   }
}

class Product{
   int id;
   String name,grade;
   double salesPrice,purchasedPrice;
   double allowed_sales_price = 0.0;
   Product(int id,String name, double purchasedPrice,double salesPrice,String grade){
      this.id = id;
      this.grade= grade;
      this.name = name;
      this.salesPrice = salesPrice;
      this.purchasedPrice = purchasedPrice;
      allowed_sales_price = ((25 * this.purchasedPrice)/100) + this.purchasedPrice;

      //Validating Data Entered

      if(!this.grade.equals("N") && !this.grade.equals("E")){
         throw new GradeMismatchException("Grade Not Allowed, Allowed Grades are 'N' and 'E' not
"+this.grade);
      }
      if(this.grade.equals("E") && this.salesPrice > allowed_sales_price){
         throw new EssentialCommodityException("'E' graded items Sales Price cannot be more than 25% of
Purchase Price.");
      }
      if(this.salesPrice < this.purchasedPrice)
         throw new PriceException("Sales Price is lesser than Purchased Price");
   }
   public String getName(){
      return this.name;
   }

   public void setName(String name){
```

```
        this.name = name;
    }

    public int getId(){
        return this.id;
    }

    public void setId(int id){
        this.id = id;
    }
    public String getGrade(){
        return this.grade;
    }

    public void setGrade(String grade){
        if(!this.grade.equals("N") || !this.grade.equals("E")){
            throw new GradeMismatchException("Grade Not Allowed, Allowed Grades are 'N' and 'E'.");
        }
        this.grade = grade;
    }
    public double getSalePrice(){
        return this.salesPrice;
    }

    public void setPurchasedPrice(double purchasedPrice){
        if(this.salesPrice < purchasedPrice)
            throw new PriceException("Sales Price is lesser than Purchased Price");
        this.purchasedPrice = purchasedPrice;
    }
    public double getPurchasedPrice(){
        return this.purchasedPrice;
    }

    public void setSalePrice(double salesPrice){
        if(this.purchasedPrice > salesPrice){
            throw new PriceException("Sales Price is lesser than Purchased Price");
        }
        this.salesPrice = salesPrice;
    }

    @Override
    public String toString(){
        return String.format("%-5s %-20s %-10s %-10s %-5s",
this.id,this.grade,this.name,this.salesPrice,this.purchasedPrice);
    }

    @Override
    public int hashCode() {
        return super.hashCode();
    }

    @Override
```

```
   public boolean equals(Object obj) {
      return super.equals(obj);
   }
}
class Main{
   public static void main(String[] args) {
      ArrayList<String> non_dup_ids = new ArrayList<>();
      ArrayList<String> valid_items = new ArrayList<>();
      ArrayList<String> items = new ArrayList<>();
      Scanner read = new Scanner(System.in);
      System.out.print("Enter number of items you want to Add :");
      int n = read.nextInt();
      read.nextLine();
      for(int i = 0 ; i < n;i++){
         String detail = read.nextLine();
         items.add(detail);
      }
      for(String detail:items){
         String[] details = detail.split(",");

         int flag = 0;
         if(non_dup_ids.contains(details[0])) continue;

         //Checking Validaity of items

         try{
            new Product(Integer.parseInt(details[0]), details[1], Double.parseDouble(details[2]),
Double.parseDouble(details[3]),details[4]);
         }catch(RuntimeException re){
            System.out.println("Item having id "+ details[0] +" "+re);
            flag = 1;
         }
         if(flag == 0){
            valid_items.add(details[1]);
         }
         non_dup_ids.add(details[0]);
      }
    System.out.println("Non Duplicate and valid items are :"+valid_items);
   }
}
```

Input :

Enter number of items you want to add :2
1,salt,22,28,E
2,Biryani Masala,45,55,N

Output:

Item having id 1 EssentialCommodityException: 'E' graded items Sales Price cannot be more than 25% of Purchase Price.
Non Duplicate and valid items are :[Biryani Masal]