Codes of three classes of the NextGen POS in C++

This code defines a simple case; it is not meant to illustrate a robust, fully developed C++ program with synchronization, exception handling, and so on.

Class ProductSpecification

```
class ProductSpecification
{
   private:
       ItemID id;
       Money price;
       string specification;
   public:
      ProductSpecification( const ItemID &id, const Money &price, const string &spec )
      {
        this->id = id;
        this->price = price;
        specification = spec;
      }
    const ItemID & getItemId() { return id; }
    const Money & getPrice() { return price; }
    const string & getSpecification() { return specification; }
};
```

Class Sale

```
class Sale
{
  private:
      vector <SalesLineItem*> lineitems;
      Date date;
      bool isComplete;
      Payment *payment;

public:
    Sale(){
       isComplete = false;
    }
    Money & getBalance()
    {
       return payment->getAmount().minus(getTotal());
    }
    void becomeComplete() { isComplete = true; }
    bool isComplete() { return isComplete; }
    void makeLineItem( const ProductSpecification *spec, int amount )
    {
         SalesLineItem *sli = new SalesLineItem(spec, amount);
         lineitems.push_back ( sli );
    }

   Money getTotal()
  {
        Money total;
        for(unsigned int j=0; j<lineitems.size(); j++){
           SalesLineItem *sli = lineitems[j];
           total.plus( sli->getSubTotal() );
        }
       return total;
   }

   void makePayment ( Money & cache )
   {
       payment = new Payment (cache);
   }

   ~Sale ()
   {
      delete payment;
   }
};           // End of class Sale
```

Class Register

```
class Register
{
   private:
      ProductCatalog * catalog;
      Sale * currentSale;
  public:
    Register( ProductCatalog * catalog )
    {
       this->catalog = catalog;
    }

    void makeNewSale()
     {
        currentSale = new Sale();
     }
    void enterItem( ItemID & id, int amount )
    {
        if (currentSale !=NULL) {
          ProductSpecification *spec = catalog->getSpecification(id);
          currentSale->makeLineItem(spec, amount);
        }
        else   ….  // exception
    }

void endSale()          // if (currentSale !=NULL) must be checked
    {
       currentSale-> becomeComplete();
    }
void makePayment( Money & cache )
   {
      currentSale->makePayment(cache);
   }
};        // End of class Register
```