SelfServiceCashRegister

May 22, 2014

Contents

1 BarcodeScanner

```
class BarcodeScanner

instance variables
 enabled: bool;

operations

 public BarcodeScanner:() ==> BarcodeScanner
 BarcodeScanner() ==
 (
  enabled := false;
 );


 public Enable:bool ==> ()
 Enable(enable) ==
  enabled := enable;


 public ScanBarcode: CashRegister`Barcode ==> ()
 ScanBarcode(barcode) ==
  if enabled then
  CR`cashRegister.AddProduct(barcode);

end BarcodeScanner
```

Function or operation	Coverage	Calls
BarcodeScanner: 7	100.0%	2410
Enable: 13	100.0%	2428
ScanBarcode: 17	100.0%	7254
BarcodeScanner.vdmpp	100.0%	12092

2 CR

```
class CR

instance variables
 public static cashRegister: CashRegister := new CashRegister();
  public static cabTerm: CoinAndBanknoteTerminal := new CoinAndBanknoteTerminal();
 public static ccTerm: CreditCardTerminal := new CreditCardTerminal();
 public static scanner: BarcodeScanner := new BarcodeScanner();
 public static screen: TouchScreen := new TouchScreen();

end CR
```

Function or operation	Coverage	Calls
CR.vdmpp	100.0%	0

3 CashRegister

```
class CashRegister

types
 public Products = map Barcode to ProductInfo;
 public ProductInfo::  Price : nat1
            WeightRange : nat1 * nat1 --Not used yet
            Name : seq of char;
 public Barcode = nat1;

values
 prices : seq of nat1 = [ price | price in set {1,...,100} & price mod 10 = 0];
 weigths : seq of (nat1*nat1) = [ mk_(10*x, 20*x) | x in set elems prices];
 names : seq of (seq of char) = ["Bread","Milk","Honey","Ham","Touthpaste","Toilet paper","Beer","Pasta","Veal Beef","Ketchup"];

 public products : CashRegister`Products = {code |-> mk_CashRegister`ProductInfo(prices(code),weigths(code),names(code))
          | code in set {1,...,10}};

instance variables
 public databaseProducts: Products;
  inv databaseProducts <> {|->} and
```

```
   forall x in set rng databaseProducts &
    x.Name <> "";

 basketProducts: seq of ProductInfo;
 totalPrice : nat;
  inv totalPrice = TotalPrice(basketProducts);

operations

 public CashRegister: () ==> CashRegister
 CashRegister() ==
  (
   databaseProducts := products;
   basketProducts := [];
   totalPrice := 0;
  )
  pre products <> {|->};


 public AddProduct: Barcode ==> ()
 AddProduct(bar) ==
  (
   if bar in set dom databaseProducts then
   (
    atomic
    (
     basketProducts := basketProducts ^ [databaseProducts(bar)];
     totalPrice := totalPrice + databaseProducts(bar).Price;
    );
   )
   else
   (
    IO`print("Barcode is not valid\n");
   )
  );


 public AddMultiple : nat1 ==> ()
 AddMultiple(number) ==
  let prod = basketProducts(len basketProducts)
   in
    atomic
    (
      basketProducts := basketProducts ^ [prod|x in set {1,...,number-1}]
      totalPrice := totalPrice + prod.Price*(number-1);
    )
  pre len basketProducts <> 0;


 public EmptyBasket: () ==> ()
 EmptyBasket() ==

  atomic
  (
   basketProducts := [];
   totalPrice := 0;
  );

 public Pay: (PaymentComponent) ==> ()
 Pay(component) ==

  (
   if component.Pay(totalPrice) then
   (
    IO`print("\nPayment receipt:\n");
    PrintReceipt(basketProducts);
    EmptyBasket();
    CR`scanner.Enable(false);
   )
  );

 PrintReceipt: seq of ProductInfo ==> ()
 PrintReceipt(prods) ==

  if (len prods = 0) then
   return
  else
   let prod = hd prods
    in
    (
     IO`print(prod.Name);
     IO`print(" : ");
     IO`print(prod.Price);
     IO`print(" DKK\n");
     PrintReceipt(tl prods);
    )

functions
 TotalPrice : seq of ProductInfo -> nat
 TotalPrice(prods) ==

  if (len prods = 0) then
   0
  else
   (hd prods).Price + TotalPrice(tl prods)
  measure CardPrice;

 CardPrice : seq of ProductInfo -> nat
 CardPrice(prods) ==

  len prods;

end CashRegister
```

Function or operation	Coverage	Calls
AddMultiple: 55	100.0%	2405
AddProduct: 38	100.0%	7254
CardPrice: 113	100.0%	83674
CashRegister: 29	100.0%	2410
EmptyBasket: 69	100.0%	2414
Pay: 77	100.0%	18
PrintReceipt: 89	100.0%	58
TotalPrice: 105	100.0%	83674
CashRegister.vdmpp	100.0%	181907

4 CoinAndBanknoteTerminal

```
class CoinAndBanknoteTerminal is subclass of PaymentComponent

instance variables
```

```
 balance : nat;

operations

 public CoinAndBanknoteTerminal: () ==> CoinAndBanknoteTerminal
  CoinAndBanknoteTerminal() ==
   balance :=0;


 public PutInMoney: nat1 ==> ()
  PutInMoney(amount) ==
   balance := balance + amount;


 public RetreiveMoney: () ==> ()
  RetreiveMoney() ==
  (
   IO`print("Giving back money: ");
   IO`print(balance);
   IO`print(" DKK\n");

   balance := 0;
  )
  pre balance > 0;



 public Pay: nat ==> bool
  Pay(sum) ==
   let enough = sum <= balance
      in
    if enough then
    (
     IO`print("Paying with cash: ");
     IO`print(sum);
     IO`print(" DKK\n");
     balance := balance - sum;
     RetreiveMoney();
     return true;
    ) else
    (
     IO`print("Insufficient funds.\n");
     return false;
    )
  pre sum <> 0;

end CoinAndBanknoteTerminal
```

Function or operation	Coverage	Calls
CoinAndBanknoteTerminal: 7	100.0%	2410
Pay: 27	100.0%	14
PutInMoney: 11	100.0%	10
RetreiveMoney: 15	100.0%	5
CoinAndBanknoteTerminal.vdmpp	100.0%	2439

5 CreditCardTerminal

```
class CreditCardTerminal is subclass of PaymentComponent

operations

 public CreditCardTerminal : () ==> CreditCardTerminal
 CreditCardTerminal() ==
  skip;


 public Pay : nat ==> bool
 Pay(sum) ==
 (
 --Emulate wrong pin, not enough money etc.
  if MATH`rand(100) < 10 then
  (
   IO`print("Wrong PIN or insufficient funds.\n");
   return false;
  ) else
  (
   IO`print("Paying with credit card: ");
   IO`print(sum);
   IO`print(" DKK\n");
   return true;
  )
 )
 pre sum <> 0;
end CreditCardTerminal
```

Function or operation	Coverage	Calls
CreditCardTerminal: 4	100.0%	2410
Pay: 8	78.2%	4
CreditCardTerminal.vdmpp	79.1%	2414

6 Environment

```
class Environment

instance variables
  scanner: BarcodeScanner;
  screen: TouchScreen;
  cabTerm: CoinAndBanknoteTerminal;

operations

 public Environment:() ==> Environment
 Environment() ==
  (
   scanner := CR`scanner;
   screen := CR`screen;
   cabTerm := CR`cabTerm;

  );


 public Run:() ==> ()
 Run() ==
 (
 -- Standard run with cash:
  screen.StartPayment();

  scanner.ScanBarcode(2);
  screen.AddMultiple(3); -- Add three products with barcode 2
  scanner.ScanBarcode(4);
  scanner.ScanBarcode(7);
  scanner.ScanBarcode(11); --Invalid barcode

  cabTerm.PutInMoney(100); -- not enough
  screen.PayCash();
  cabTerm.PutInMoney(200);
  screen.PayCash();

 -- Cancel payment run:
  screen.StartPayment();
  scanner.ScanBarcode(4);
  scanner.ScanBarcode(7);
  screen.CancelPayment();

 -- Standard run with credit card:
  screen.StartPayment();
  scanner.ScanBarcode(1);
  scanner.ScanBarcode(5);
  scanner.ScanBarcode(6);
  scanner.ScanBarcode(8);
  scanner.ScanBarcode(9);
  scanner.ScanBarcode(10);

  screen.PayCredit();
 )

traces

PayWithCredit:
 let myBarcodes in set dom CR`cashRegister.databaseProducts
 in
  let myBarcodes2 in set dom CR`cashRegister.databaseProducts
  in
  (
   screen.StartPayment();
   scanner.ScanBarcode(myBarcodes){1};
   scanner.ScanBarcode(myBarcodes2){1,3};
   screen.PayCredit()
  );


PayWithCash:
 let myBarcodes in set dom CR`cashRegister.databaseProducts
 in
  let myBarcodes2 in set dom CR`cashRegister.databaseProducts
  in
  (
   screen.StartPayment();
   scanner.ScanBarcode(myBarcodes){1};
   scanner.ScanBarcode(myBarcodes2){1,3};
   cabTerm.PutInMoney(500);
   screen.PayCash()
  );

AddingAndEmptyingBasket:
 let myBarcodes in set dom CR`cashRegister.databaseProducts
 in
  let myBarcodes2 in set dom CR`cashRegister.databaseProducts
  in
   let mult in set {1,...,4}
   in
   (
    screen.StartPayment();
    scanner.ScanBarcode(myBarcodes){1};
    screen.AddMultiple(mult);
    scanner.ScanBarcode(myBarcodes2){1,3};
    screen.CancelPayment()
   );

preconditionFail:
 let mult = 1
  in
  (
   screen.StartPayment();
   screen.AddMultiple(mult)
  );

end Environment
```

Function or operation	Coverage	Calls
Environment: 9	100.0%	4820
Run: 18	100.0%	8
Environment.vdmpp	48.9%	4828

7 PaymentComponent

```
class PaymentComponent

operations

 public PaymentComponent : () ==> PaymentComponent
 PaymentComponent() ==
 skip;


 public Pay : nat ==> bool
 Pay(-) == is subclass responsibility;

end PaymentComponent
```

Function or operation	Coverage	Calls
Pay: 8	100.0%	11
PaymentComponent: 4	100.0%	4820
PaymentComponent.vdmpp	100.0%	4831

8 TouchScreen

```
class TouchScreen


operations

 public TouchScreen : () ==> TouchScreen
 TouchScreen() ==
  skip;


 public StartPayment : () ==> ()
 StartPayment() ==
  CR`scanner.Enable(true);


 public PayCash : () ==> ()
 PayCash() ==
  CR`cashRegister.Pay(CR`cabTerm);


 public PayCredit : () ==> ()
 PayCredit() ==
  CR`cashRegister.Pay(CR`ccTerm);


 public AddMultiple : nat1 ==> ()
 AddMultiple(number) ==
  CR`cashRegister.AddMultiple(number);


 public CancelPayment: () ==> ()
 CancelPayment() ==
  CR`cashRegister.EmptyBasket();

end TouchScreen
```

Function or operation	Coverage	Calls
AddMultiple: 21	100.0%	3606
CancelPayment: 25	100.0%	2405
PayCash: 13	100.0%	14
PayCredit: 17	100.0%	4
StartPayment: 9	100.0%	2419
TouchScreen: 5	100.0%	2410
TouchScreen.vdmpp	100.0%	10858

9 World

```
class World

instance variables
  public static env: Environment := new Environment();


operations
```

```
 public World:() ==> World
 World() ==
  env := new Environment();



 public Run:() ==> ()
 Run() ==
  env.Run();


end World
```

Function or operation	Coverage	Calls
Run: 13	100.0%	8
World: 8	100.0%	8
World.vdmpp	100.0%	16