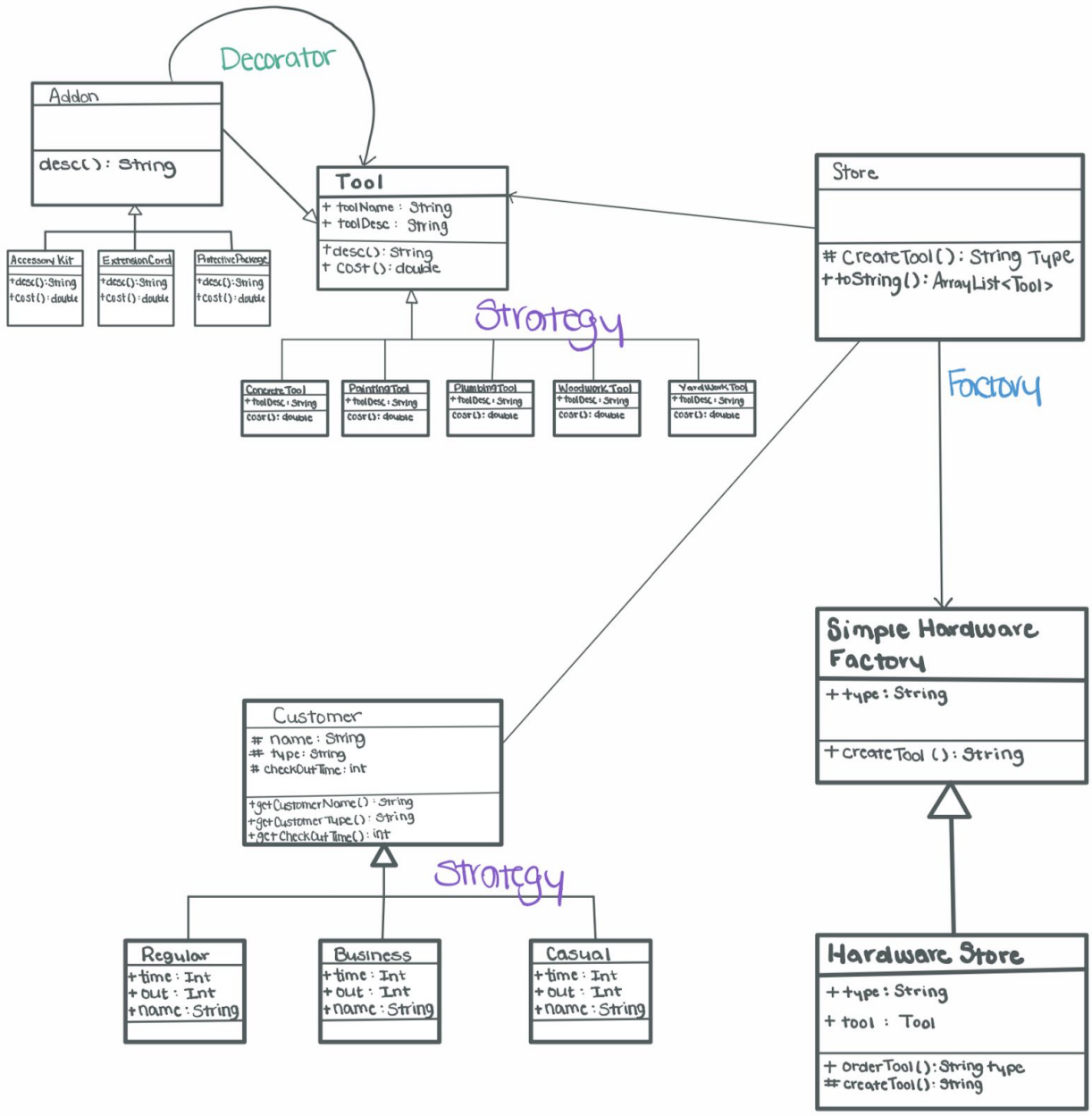
Nimra Sharnez
Khalid Aldawood



Design & Assumptions

We are thinking about using the three patterns:
Factory: For the hardware store
Strategy: For the Customers and Tools
Decorator: For the accessories we will use on the tools.

We also considered using the Observer Pattern for announcing the logging of items customers are checking out but will instead be going with a Hashmap to keep track of each customer's

checkout record paired with an array list called Inventory that will keep track of the items in the store's inventory.
The main method will be held in the Concrete Implementation of the Store to allow customers to checkout items that were created through the createTool method.
Lastly, we intend to delegate small details of checking out items. This will include a string that will be returned to print out the information of the item checkout out. It will take into account any decorators added and the total price based off of the daily rate and the days checkouted out.
Another method in this class (called infroGenerator) will include the code to generate a cart by seeing what is available in the inventory and adding it to the cart of the customer that entered the store and removing it from the store inventory list.
We think that to simulate the days and the amount of items that will allow a customer to enter a store will be handled in main through a for loop.

```
//Business.java
package HardwareStoreSimulation.Customer;

public class Business extends Customer {

   public Business(String name) {
         super(name);
         this.type = "Business";
         this.checkoutTime = 7;
         this.toolsOut = 3;
   }
}

//Casual.java
package HardwareStoreSimulation.Customer;

import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class Casual extends Customer {
         List<Integer> givenList = Arrays.asList(1, 2);


         Random rand = new Random();
         int time = givenList.get(rand.nextInt(givenList.size()));
         int out = givenList.get(rand.nextInt(givenList.size()));
```

```
    public Casual(String name) {
            super(name);
            this.type= "Casual";
            this.checkoutTime = time; // time * tool.price = how much pay upfront
            this.toolsOut = out;

    }
}
```

```
//Regular.java
package HardwareStoreSimulation.Customer;

import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class Regular extends Customer{

            List<Integer> givenList = Arrays.asList(3, 4, 5);
            List<Integer> outTools = Arrays.asList(1, 2, 3);

            Random rand = new Random();
            int time = givenList.get(rand.nextInt(givenList.size()));
            int out = outTools.get(rand.nextInt(outTools.size()));

      public Regular(String name) {
            super(name);
            this.type = "Regular";
            this.checkoutTime = time;
            this.toolsOut = out;
      }

    }

//Customer.java
package HardwareStoreSimulation.Customer;
```

Nimra Sharnez
Khalid Aldawood

```
public class Customer {

        protected String name;
    protected String type;
        protected int checkoutTime;
        protected int toolsOut;

        public Customer(String name) {
        this.name = name;
        }
        public String getCustomerName() {
        return this.name;
        }
        public String getCustomerType() {
        return this.type;
        }

        public int getCheckoutTime() {
          return this.checkoutTime;
        }

        public int toolsOut() {
          return this.toolsOut;
        }
}


//Tool.java
package HardwareStoreSimulation.Tool;


//The product of the factory; Tool!
//defined as an abstract class with some implementations that could be overridden
public abstract class Tool {

        public String toolName;
        public String toolDesc = "Unknown Tool";

        public String desc() {
         return toolDesc;
        }
```

```
    public abstract double cost();
    public abstract int id();


}

//YardworkTool.java
package HardwareStoreSimulation.Tool;

public class YardworkTool extends Tool{

        public YardworkTool() {
        toolDesc = "YardworkTool";

        }

        public double cost() {
         return 3.00;
        }

        public int id() {
         return 3;
        }

}

//WoodworkTool.java
package HardwareStoreSimulation.Tool;

public class WoodworkTool extends Tool{


        public WoodworkTool() {
        toolDesc = "WoodworkTool";

        }

        public double cost() {
         return 8.00;
        }

        public int id() {
         return 2;
```

```
        }
}

//PlumbingTool.java
package HardwareStoreSimulation.Tool;

public class PlumbingTool extends Tool {


        public PlumbingTool() {
        toolDesc = "PlumbingTool";

        }

        public double cost() {
         return 6.00;
        }

        public int id() {
         return 4;
        }


}

//PaintingTool.java
package HardwareStoreSimulation.Tool;

public class PaintingTool extends Tool {

        public PaintingTool() {
        toolDesc = "PaintingTool";

        }

        public double cost() {
         return 2.00;
        }

        public int id() {
         return 1;
        }
```

```
}

//ConcreteTool.java
package HardwareStoreSimulation.Tool;

public class ConcreteTool extends Tool {

        public ConcreteTool() {
        toolDesc = "ConcreteTool";

        }

        public double cost() {
         return 9.00;
        }

        public int id() {
         return 5;
        }


}

//AccessoryKit.java
package HardwareStoreSimulation.Tool.Decorators;

import HardwareStoreSimulation.Tool.Tool;

public class AccessoryKit extends Addon {
   Tool tool;

   public AccessoryKit(Tool tool) {
          this.tool = tool;
   }

   public String desc() {
          return tool.desc() + ", with AccessoryKit";
   }

   public double cost() {
          return tool.cost() + 10.0;
   }
```

```
    @Override
    public int id() {
        // TODO Auto-generated method stub
        return 0;
    }
}

//Addon.java
package HardwareStoreSimulation.Tool.Decorators;

import HardwareStoreSimulation.Tool.Tool;

public abstract class Addon extends Tool{

    public abstract String desc();
}

//ExtensionCord.java
package HardwareStoreSimulation.Tool.Decorators;

import HardwareStoreSimulation.Tool.Tool;

public class ExtensionCord extends Addon{
    Tool tool; //instantiate ExtensionCord with reference to Tool.

    public ExtensionCord(Tool tool) {
        this.tool = tool;
    }

    public String desc() {
        return tool.desc() + ", with ExtensionCord";
    }

    public double cost() {
        return tool.cost() + 1.0;
    }

    @Override
    public int id() {
        // TODO Auto-generated method stub
        return 0;
    }
```

```
}

//ProtectiveGearPackage.java
package HardwareStoreSimulation.Tool.Decorators;

import HardwareStoreSimulation.Tool.Tool;

public class ProtectiveGearPackage extends Addon {

    Tool tool;

    public ProtectiveGearPackage(Tool tool) {
        this.tool = tool;
    }

    public String desc() {
        return tool.desc() + ", with ProtectiveGearPackage";
    }

    public double cost() {
        return tool.cost() + 7.0;
    }

    @Override
    public int id() {
        // TODO Auto-generated method stub
        return 0;
    }
}

//HardwareStore.java
package HardwareStoreSimulation;


import java.util.ArrayList;

import HardwareStoreSimulation.Tool.Tool;

//the client of factory, goes through Simple Hardware Factory to get instance of tool
public abstract class HardwareStore {

    public Tool orderTool(String type) { //has no idea what type of tool we are creating
```

```
        Tool tool;

        tool = createTool(type);

        return tool;
    }
    protected abstract Tool createTool(String type); //all resp. for instantiating Tools has been
moved into a method that acts as a factory
    public String toString(ArrayList<Tool> list) {
        // TODO Auto-generated method stub
        return null;
    }
}



//InfoGenerator.java
package HardwareStoreSimulation;
import java.util.Random;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;

import HardwareStoreSimulation.Customer.Customer;
import HardwareStoreSimulation.Tool.Tool;
import HardwareStoreSimulation.Tool.Decorators.AccessoryKit;
import HardwareStoreSimulation.Tool.Decorators.ExtensionCord;
import HardwareStoreSimulation.Tool.Decorators.ProtectiveGearPackage;

public class InfoGenerator {



    //RETURNS THE INFO
    public static String info(Tool tool, int ext, int days) {
        //ext is the number of extensions they want to add
        //need to make a way to add 0->6 extensions added to cost

        Tool t= tool;


        double cost = t.cost();
```

```
        String desc = t.desc();

        if (ext == 0) {

        }
        else if (ext == 1) {
                t = new AccessoryKit(t);
        }
        else if (ext == 2) {
                t = new AccessoryKit(t);
                t = new ExtensionCord(t);
        }
        else if (ext == 3) {
                t = new AccessoryKit(t);
                t = new ExtensionCord(t);
                t = new ProtectiveGearPackage(t);
        }
        else if (ext == 4) {
                t = new ExtensionCord(t);
        }
        else if (ext == 5) {
                t = new ExtensionCord(t);
                t = new ProtectiveGearPackage(t);
        }
        else if (ext == 6) {
                t = new ProtectiveGearPackage(t);
        }
        else if (ext == 7) {
                t = new AccessoryKit(t);
                t = new ProtectiveGearPackage(t);
        }
        else {

        }

        cost = t.cost();
        desc = t.desc();

        return cost*days + " dollars for " + desc + " today because it is rented out for " + days + "
days.";

    }
```

Nimra Sharnez
Khalid Aldawood

```
//Removes a tool out of inventory, puts into shopper's cart, update inventory
public ArrayList<Tool> cart(Customer c, ArrayList<Tool> Inventory){

	ArrayList<Tool> cart = new ArrayList<Tool>(); //empty cart

	Random rand = new Random();

	//randomly pick item from inventory 0 ->24
	int n = rand.nextInt(Inventory.size());


	//add item to cart
	cart.add(Inventory.get(n));
	 //remove item from inventory
	Inventory.remove(n);



	return cart;

}

//Removes tool from cart and puts back in inventory, update cart and inventory
public ArrayList<Tool> ret(ArrayList<Tool> cart, ArrayList<Tool> Inventory){


	for (int i=0; i<cart.size(); i++) {
		Inventory.add(cart.get(i));
	}

	cart = new ArrayList<Tool>();
	return cart;

}

//finds Customer associated with cart
public static <K, V> K getKey(Map<K, V> map, V value) {
	for (K key : map.keySet()) {
		if (value.equals(map.get(key))) {
			return key;
		}
	}
	return null;
```

```
    }


}

//SimpleHardwareFactory.java
package HardwareStoreSimulation;

import java.util.List;

import HardwareStoreSimulation.Tool.PaintingTool;
import HardwareStoreSimulation.Tool.PlumbingTool;
import HardwareStoreSimulation.Tool.Tool;
import HardwareStoreSimulation.Tool.WoodworkTool;
import HardwareStoreSimulation.Tool.YardworkTool;


public class SimpleHardwareFactory {
    public Tool createTool(String type) { //factory method
            Tool tool = null;

            return tool;
    }

}


//Store.java

package HardwareStoreSimulation;


import HardwareStoreSimulation.Customer.Business;

import HardwareStoreSimulation.Customer.Casual;
import HardwareStoreSimulation.Customer.Customer;
import HardwareStoreSimulation.Customer.Regular;
import HardwareStoreSimulation.Tool.ConcreteTool;
import HardwareStoreSimulation.Tool.PaintingTool;
import HardwareStoreSimulation.Tool.PlumbingTool;
import HardwareStoreSimulation.Tool.Tool;
import HardwareStoreSimulation.Tool.WoodworkTool;
```

Nimra Sharnez
Khalid Aldawood

```
import HardwareStoreSimulation.Tool.YardworkTool;
import HardwareStoreSimulation.Tool.Decorators.AccessoryKit;
import HardwareStoreSimulation.Tool.Decorators.ExtensionCord;
import HardwareStoreSimulation.Tool.Decorators.ProtectiveGearPackage;
import java.util.*;
public class Store extends HardwareStore{

    @Override
    protected
    Tool createTool(String item) {

            if (item.equals("paint")) {
                    return new PaintingTool();
            }
            else if (item.equals("wood")) {
                    return new WoodworkTool();
            }
            else if (item.equals("yard")) {
                    return new YardworkTool();
            }
            else if (item.equals("plumbing")) {
                    return new PlumbingTool();
            }
            else if (item.equals("concrete")) {
                    return new ConcreteTool();
            }
            else return null;


    }

    @Override
    public String toString(ArrayList<Tool> list) {
            String result = ", ";
            for (int i = 0; i < list.size(); i++) {
            result += " " + list.get(i);
            }
            return result;
    }

    public static void main(String[] args) {

            //creating instance of store
```

```
HardwareStore bobsStore = new Store();

//4 Regular
Customer Brad = new Regular("Brad");
Customer Paul = new Regular("Paul");
Customer Larry = new Regular("Larry");
Customer Santos = new Regular("Santos");
//4 Business
Customer Rhonda = new Business("Rhonda");
Customer Harry = new Business("Harry");
Customer Florence = new Business("Florence");
Customer Gary = new Business("Gary");
//4 Casual
Customer Kevin = new Casual("Kevin");
Customer Dan = new Casual("Dan");
Customer Norman = new Casual("Norman");
Customer Zach = new Casual("Zach");

ArrayList<Tool> Inventory = new ArrayList<Tool>();
//5 Painting Tools
Tool p0 = bobsStore.orderTool("paint");
p0.toolName = p0.toolDesc + " 1";
Inventory.add(p0);
Tool p1 = bobsStore.orderTool("paint");
p1.toolName = p1.toolDesc + " 2";
Inventory.add(p1);
Tool p2 = bobsStore.orderTool("paint");
p2.toolName = p2.toolDesc + " 3";
Inventory.add(p2);
Tool p3 = bobsStore.orderTool("paint");
p3.toolName = p3.toolDesc + " 4";
Inventory.add(p3);
Tool p4 = bobsStore.orderTool("paint");
p4.toolName = p4.toolDesc + " 5";
Inventory.add(p4);

//4 Woodworking Tools
Tool w0 = bobsStore.orderTool("wood");
w0.toolName = w0.toolDesc + " 1";
Inventory.add(w0);
Tool w1 = bobsStore.orderTool("wood");
w1.toolName = w1.toolDesc + " 2";
Inventory.add(w1);
```

```
Tool w2 = bobsStore.orderTool("wood");
w2.toolName = w2.toolDesc + " 3";
Inventory.add(w2);
Tool w3 = bobsStore.orderTool("wood");
w3.toolName = w3.toolDesc + " 4";
Inventory.add(w3);

//5 Yardworking Tools
Tool y0 = bobsStore.orderTool("yard");
y0.toolName = y0.toolDesc + " 1";
Inventory.add(y0);
Tool y1 = bobsStore.orderTool("yard");
y1.toolName = y1.toolDesc + " 2";
Inventory.add(y1);
Tool y2 = bobsStore.orderTool("yard");
y2.toolName = y2.toolDesc + " 3";
Inventory.add(y2);
Tool y3 = bobsStore.orderTool("yard");
y3.toolName = y3.toolDesc + " 4";
Inventory.add(y3);
Tool y4 = bobsStore.orderTool("yard");
y4.toolName = y4.toolDesc + " 5";
Inventory.add(y4);

//5 Plumbing Tools
Tool pl0 = bobsStore.orderTool("plumbing");
pl0.toolName = pl0.toolDesc + " 1";
Inventory.add(pl0);
Tool pl1 = bobsStore.orderTool("plumbing");
pl1.toolName = pl1.toolDesc + " 2";
Inventory.add(pl1);
Tool pl2 = bobsStore.orderTool("plumbing");
pl2.toolName = pl2.toolDesc + " 3";
Inventory.add(pl2);
Tool pl3 = bobsStore.orderTool("plumbing");
pl3.toolName = pl3.toolDesc + " 4";
Inventory.add(pl3);
Tool pl4 = bobsStore.orderTool("plumbing");
pl4.toolName = pl4.toolDesc + " 5";
Inventory.add(pl4);

//5 Concrete Tools
Tool c0 = bobsStore.orderTool("concrete");
```

```
        c0.toolName = c0.toolDesc + " 1";
        Inventory.add(c0);
        Tool c1 = bobsStore.orderTool("concrete");
        c1.toolName = c1.toolDesc + " 2";
        Inventory.add(c1);
        Tool c2 = bobsStore.orderTool("concrete");
        c2.toolName = c2.toolDesc + " 3";
        Inventory.add(c2);
        Tool c3 = bobsStore.orderTool("concrete");
        c3.toolName = c3.toolDesc + " 4";
        Inventory.add(c3);
        Tool c4 = bobsStore.orderTool("concrete");
        c4.toolName = c4.toolDesc + " 5";
        Inventory.add(c4);

//      //Customer Carts
//      ArrayList<Tool> Bcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Pcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Lcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Scart = new ArrayList<Tool> ();
//      ArrayList<Tool> Rcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Hcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Fcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Gcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Kcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Dcart = new ArrayList<Tool> ();
//      ArrayList<Tool> Ncart = new ArrayList<Tool> ();
//      ArrayList<Tool> Zcart = new ArrayList<Tool> ();

        //Checked Out Hash Map
        HashMap<Customer, ArrayList<Tool>> Out = new HashMap<>();
        Out.put(Brad, new ArrayList<Tool>());
        Out.put(Paul, new ArrayList<Tool>());
        Out.put(Larry, new ArrayList<Tool>());
        Out.put(Santos, new ArrayList<Tool>());
        Out.put(Rhonda, new ArrayList<Tool>());
        Out.put(Harry, new ArrayList<Tool>());
        Out.put(Florence, new ArrayList<Tool>());
        Out.put(Gary, new ArrayList<Tool>());
        Out.put(Kevin,  new ArrayList<Tool>());
        Out.put(Dan, new ArrayList<Tool>());
        Out.put(Norman, new ArrayList<Tool>());
        Out.put(Zach, new ArrayList<Tool>());
```

```
		ArrayList<Integer> checkout = new ArrayList<Integer>();
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);
		checkout.add(0);

		//Brad.getCheckoutTime();
		//checkout.add(0, 0);


//		List<Customer> names = new ArrayList<Customer>(Out.keySet());

		Random r = new Random();
////////////////////////////////////////////////////////////////////////////////////

		//Instantiating Information Generator
		InfoGenerator generate = new InfoGenerator();

		//each day a random customer visits the store
		for (int days = 1; days<=35; days++) {




			int counter12 = 0;
			for (Customer key : Out.keySet()) {
				System.out.println("ACTIVE RENTALS: " + key.getCustomerName() + "
checked out: " + Out.get(key) + " checked out.");

				if (checkout.get(counter12).equals(0)) { //has no checkout time


					if (!Out.get(key).isEmpty()) { //does cart have items?
```

```
                                        for (int k=0; k<Out.get(key).size(); k++) {
                                                System.out.println();
                                                System.out.println( Out.get(key).get(k).toolName +
" was returned.");
                                                System.out.println();

                                                //
                                        }
                                        generate.ret(Out.get(key), Inventory);
                                        Out.put(key,new ArrayList<Tool>());
                                }


                        }
                counter12++;
                }
                System.out.println();
                System.out.println("Store Inventory: " + Inventory.size() + " items: " + Inventory);
                System.out.println();
                //DECREMENT EVERY ONE EVERYDAY

                for (int h=0; h<checkout.size(); h++) {
                        if (checkout.get(h).equals(0)) {

                        }
                        else {
                                int curr = checkout.get(h);
                                curr = curr-1;
                                checkout.add(h, curr);
                                checkout.remove(h+1);
                        }
                }


                System.out.println("Day " + days);
                System.out.println();


        if (Inventory.size()>3) { //if there are more than 3 tools
        Customer customer = new Customer("Abu");
        ArrayList<Tool> cart = new ArrayList<Tool>();
```

Nimra Sharnez
Khalid Aldawood

```
//random cart
int keyC=0;
int r1 = r.nextInt(Out.size());
Out.get(r1);
for (Customer key : Out.keySet()) {
	//System.out.println(key.getCustomerName());
	if(keyC == r1) {

		customer = key;


	}


		keyC++;
	}
//System.out.println("R1: " + r1 + customer.getCustomerName());
cart = Out.get(customer);




cart = generate.cart(customer, Inventory);


Out.put(customer,cart);




int item = cart.size()-1;
int checkoutt = customer.getCheckoutTime();

if(checkout.get(r1).equals(0)) { //if they don't have a checkouttime


	checkout.add(r1, checkoutt);
	checkout.remove(r1+1);

}

else { //if they already have checkouttime so do nothing
```

```
                checkoutt = checkout.get(r1);


        }



        int r2 = new Random().nextInt(8);

        String info = generate.info(cart.get(item), r2, checkoutt);


   System.out.println(customer.getCustomerType() + ": " + customer.getCustomerName() + " is
renting '"
        + cart.get(item).toolName + "'. " + " The daily rate is: $"
        + cart.get(item).cost() + ". The Store made " + info);
   System.out.println();
   //System.out.println(cart);









//          System.out.println(customer.getCustomerType() + ": " + customer.getCustomerName()
+ " is returning '" +
//          cart.get(item).toolName + "'. ");
//          cart = generate.ret(cart, Inventory);
//


}



        else {
                //System.out.println("Inventory Size: " + Inventory.size());
        }
```

```
}
}
}
```