import java.util.Scanner;  
  
/\*\* ISPBilling makes bills based on ISPCharges  
 \*  
 \* @author Nancy Harris  
 \* @version V2 11/2013  
 \*/  
public class ISPBillingV2  
{  
 private static Scanner keyboard;   
   
 /\*\* main creates the bill  
 \*/  
 public static void main(String[] args)  
 {  
 char pkg;  
 double hours;  
 double savings;  
 ISPCharge charge;  
   
 keyboard = new Scanner(System.in);  
 printHeading();   
   
 pkg = readPackage("Enter the package code (A, B, or C): ");  
 hours = readHours("Enter the number of hours used: ");  
   
 charge = new ISPCharge (pkg, hours);  
   
 printBill(charge);  
   
 }  
   
 /\*\* checkSavings will perform the output for calculating  
 \* the savings with other packages  
 \*/  
 public static void checkSavings(ISPCharge charge)  
 {   
 double savings;  
   
 if (charge.saveWithB() || charge.saveWithC())  
 {  
 System.out.println();  
   
 if (charge.saveWithB())  
 {  
 savings = charge.savingsWithB();  
 System.out.printf("You could have saved $%,.2f with package B." +   
 " Call 1-888-555-1234 for more information.\n", savings);  
 }  
 if (charge.saveWithC())  
 {  
 savings = charge.savingsWithC();  
 System.out.printf("You could have saved $%,.2f with package C." +   
 " Call 1-888-555-1234 for more information.\n", savings);  
 }  
 }  
 }  
 /\*\* readPackage prompts the user and reads in the   
 \* package code. If a bad code is entered  
 \* it returns a default.  
 \*  
 \* @param prompt The prompt to use  
 \*/  
 public static char readPackage(String prompt)  
 {  
 String pkg;  
   
 System.out.print(prompt);  
 pkg = keyboard.next();  
   
 if(!(pkg.length() == 1 && pkg.toUpperCase().charAt(0) >= 'A'   
 && pkg.toUpperCase().charAt(0) <= 'C'))  
 {  
 System.out.printf("You entered %s. Using A\n", pkg);  
 pkg = "A";  
 }  
   
 return pkg.toUpperCase().charAt(0);  
 }  
   
 /\*\* readHours reads in the hours defaulting if  
 \* bad values are read in.  
 \*  
 \* @param prompt The prompt to use  
 \*/  
 public static double readHours(String prompt)  
 {  
 double hours;  
   
 System.out.print(prompt);  
 if (keyboard.hasNextDouble())  
 {  
 hours = keyboard.nextDouble();  
 if (hours < 0)  
 {  
 System.out.printf("You entered %f. Using 0\n", hours);  
 hours = 0;  
 }  
 }  
 else  
 {  
 System.out.printf("You entered %s. Using 0\n", keyboard.nextLine());  
 hours = 0;  
 }  
 return hours;  
 }  
   
 /\*\* printHours prints the additional hours  
 \*  
 \* @param charge The charge to print  
 \*/  
 public static void printAdditional(ISPCharge charge)  
 {  
 System.out.printf("Base Charge: $%.2f\n", charge.getBase());  
 System.out.printf("Additional Hours: %.2f\n", charge.getAddtlHours());  
 }  
   
 /\*\* printHeading prints the heading  
 \*/  
 public static void printHeading()  
 {  
 System.out.println("Dukes ISP Billing");  
 System.out.println();  
 }  
 /\*\* printBill prints this bill  
 \*  
 \* @param charge The ISPCharge for this bill  
 \*/  
 public static void printBill(ISPCharge charge)  
 {  
 System.out.println();  
 System.out.println("Customer Bill");  
 System.out.println();  
 System.out.printf("Package: %s\n", charge.getPackage() );  
 System.out.printf("Hours Used: %.2f\n\n", charge.getBaseHours() + charge.getAddtlHours());  
 if(charge.needAddtlHours())  
 {   
 printAdditional(charge);  
 }  
 System.out.println();  
 System.out.printf("Total Charge: $%.2f\n", charge.calcCost());  
 System.out.printf("Tax: $%.2f\n", charge.calcTax());  
 System.out.println();  
 System.out.printf("Pay this Amount: $%.2f\n", charge.calcCost() + charge.calcTax());  
   
 checkSavings(charge);  
 }  
}

/\*\* ISPCharge represents an internet charge  
 \*  
 \* @author Nancy Harris  
 \* @version V1 10/2013  
 \*/  
public class ISPCharge  
{  
 private final double A\_CEILING = 10.0;  
 private final double A\_PRICE\_HOUR = 2.0;  
 private final double A\_PRICE\_MONTH = 9.95;  
 private final double B\_CEILING = 20.0;  
 private final double B\_PRICE\_HOUR = 1.0;  
 private final double B\_PRICE\_MONTH = 13.95;  
 private final double C\_PRICE\_MONTH = 19.95;  
 private final double TAX\_RATE = .05;  
  
 // variables describing this charge.  
 private char packageCode;  
 private double hours;  
   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
 \* The constructor sets the package and hours attributes.  
 \* @param pkg The code for the package, A, B, or C  
 \* @param hours The number of hours this month  
 \*/  
 public ISPCharge(char pkg, double hrs)  
 {  
 this.packageCode = Character.toUpperCase(pkg);  
 this.hours = hrs;  
 }  
   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* calc charge will decide which package to apply  
 \* and will return the correct cost.  
 \*  
 \* @return The charges for this month.  
 \*/  
 public double calcCost()  
 {  
 double cost;  
   
 switch (packageCode)  
 {  
 case 'A': cost = calcA(); break;  
 case 'B': cost = calcB(); break;  
 case 'C': cost = calcC(); break;  
 default: cost = 0;  
 }  
 return cost;  
 }  
  
   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* calcA calculates the charges for package A  
 \*  
 \* @return The cost for package A  
 \*/  
 public double calcA()  
 {   
 double cost;  
   
 cost = A\_PRICE\_MONTH;  
   
 if (hours > A\_CEILING)  
 {  
 cost = cost + (hours - A\_CEILING) \* A\_PRICE\_HOUR;  
 }   
   
 return cost;  
 }  
   
   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* calcB calculates the charges for package B  
 \*  
 \* @return The cost for package B  
 \*/  
 public double calcB()  
 {   
 double cost;  
   
 cost = B\_PRICE\_MONTH;  
   
 if (hours > B\_CEILING)  
 {  
 cost = cost + (hours - B\_CEILING) \* B\_PRICE\_HOUR;  
 }   
 return cost;  
 }  
   
   
   
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* calcC calculates the charges for package C  
 \*  
 \* @return The cost for package C  
 \*/  
 public double calcC()  
 {  
 return C\_PRICE\_MONTH;  
 }  
   
   
 /\*\* calcTax calculates the tax on the passed charge  
 \*  
 \* @return The tax for this charge.  
 \*/  
 public double calcTax()  
 {  
 return calcCost() \* TAX\_RATE;  
 }  
   
   
 /\*\* saveWithB calculates whether or not this   
 \* charge would be less if they were on plan B  
 \*   
 \* @return true if you can save with B, false   
 \* otherwise.  
 \*/  
 public boolean saveWithB()  
 {  
 ISPCharge b\_option;  
 boolean result;  
   
 result = false;  
 if (packageCode == 'A')  
 {  
 b\_option = new ISPCharge('B', hours);  
 result = this.calcCost() > b\_option.calcCost();  
 }  
 return result;  
 }  
   
 /\*\* saveWithC calculates whether or not this   
 \* charge would be less if they were on plan C  
 \*   
 \* @return true if there are savings with C  
 \* false otherwise  
 \*/  
 public boolean saveWithC()  
 {  
 ISPCharge c\_option;  
 boolean result;  
   
 result = false;  
 if (packageCode == 'A' || packageCode == 'B')  
 {  
 c\_option = new ISPCharge('C', hours);  
 result = this.calcCost() > c\_option.calcCost();  
 }  
 return result;  
 }  
   
 /\*\* savingsWithB calculates the savings with planB  
 \*   
 \* @return the amount of saving with B, 0 if   
 \* no savings.  
 \*/  
 public double savingsWithB()  
 {  
 ISPCharge b\_option;  
 double result;  
   
 result = 0.0;  
   
 if (saveWithB())  
 {  
 b\_option = new ISPCharge('B', hours);  
 result = b\_option.calcCost() - this.calcCost();  
 }  
 return Math.abs(result);  
 }  
   
 /\*\* savingsWithC calculates the savings if the   
 \* charge would be less if they were on plan C  
 \*   
 \* @return the amount of saving with C.  
 \*/  
 public double savingsWithC()  
 {  
 ISPCharge c\_option;  
 double result;  
   
 result = 0.0;  
 if (saveWithC())  
 {  
 c\_option = new ISPCharge('C', hours);  
 result = c\_option.calcCost() - this.calcCost();  
 }  
 return Math.abs(result);  
 }  
  
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* toString describes this charge. It should include the   
 \* package for this charge and the hours.  
 \*  
 \* @return a String representation of this package  
 \*/   
 public String toString()  
 {  
 return String.format("Package: %s\tHours: %f", this.packageCode, this.hours);  
 }  
   
 /\*\* needAddtlHours records whether or not   
 \* additional hours are needed for this   
 \* package  
 \*  
 \* @return true if we need to include additional  
 \* hours, false otherwise  
 \*/  
 public boolean needAddtlHours()  
 {  
 boolean addtl;  
   
 if (packageCode == 'A' || packageCode == 'B')  
 addtl = true;  
 else  
 addtl = false;  
 return addtl;  
 }  
 /\*\* getAddtlHours calculates the additional hours  
 \* based on this package code  
 \*   
 \* @return the additional hours  
 \*/  
 public double getAddtlHours()  
 {  
 double extra;  
 extra = 0;   
   
 if (needAddtlHours())  
 {  
 if (this.packageCode == 'A' && this.hours > this.A\_CEILING)  
 {  
 extra = this.hours - this.A\_CEILING;  
 }  
 else if (this.packageCode == 'B' && this.hours > this.B\_CEILING)  
 {  
 extra = this.hours - this.B\_CEILING;  
 }  
 }  
   
 return extra;  
 }  
 /\*\* getAddtlCharge calculates the   
 \* additional charge for this package  
 \*  
 \*@return this additional charge.  
 \*/  
 public double getAddtlCharge()  
 {  
 double extra;  
 extra = 0;   
   
 if (needAddtlHours())  
 {  
 if (this.packageCode == 'A' && this.hours > this.A\_CEILING)  
 {  
 extra = getAddtlHours() \* this.A\_PRICE\_HOUR;  
 }  
 else if (this.packageCode == 'B' && this.hours > this.B\_CEILING)  
 {  
 extra = getAddtlHours() - this.B\_PRICE\_HOUR;  
 }  
 }  
   
 return extra;  
  
 }  
 /\*\* getBase returns the base charge  
 \*  
 \* @return the base charge for this package  
 \*/  
 public double getBase()  
 {  
 double base;  
 if (packageCode == 'C')  
 base = this.C\_PRICE\_MONTH;  
 else if (packageCode == 'B')  
 base = this.B\_PRICE\_MONTH;  
 else  
 base = this.A\_PRICE\_MONTH;  
   
 return base;  
   
 }  
   
 /\*\* getBaseHours returns the base  
 \* hours for this package  
 \*  
 \* @return base hours.   
 \*/  
 public double getBaseHours()  
 {  
 double base;  
 if (packageCode == 'A')  
 base = this.A\_CEILING;  
 else   
 base = this.B\_CEILING;  
 return base;   
 }  
  
 /\*\* getPackage returns the standardized   
 \* package code  
 \*  
 \* @return this package code  
 \*/  
 public char getPackage()  
 {  
 return packageCode;  
 }  
}