This problem has been designed to help you learn multiple concepts: abstract class, interface, static variable, constants, and polymorphism.

Problem statement: Write a program to simulate dinner in which a user is asked what s/he wants to eat (Table 1). Based on the choice:

1. One of the three food-items is served with a message 'Here comes food!' followed by 'Serving X' where x is the food item being served
2. If Pizza is ordered, it is heated to a constant temperature specified in the Heatable interface as HOTSERVINGTEMPERATURE.
3. The served food item makes a certain sound while eating (Table 2)
4. A calorie counter is incremented as per the item served (Table 2)
5. The program asks the user if s/he wants more. If y, then the program repeats itself. If n then it prints Good night ! and exits.

Table 1: Console I/O

```
What would you like to eat?
1. Pizza
2. Chips
3. Ice cream
1
Here comes food!
Serving Pizza
Now its hot @ 165 degrees
Chomp Chomp
You have consumed 800 calories
Want some more (y/n)?
y
What would you like to eat?
1. Pizza
2. Chips
3. Ice cream
2
Here comes food!
Serving Chips
Crunch Crunch
You have consumed 1000 calories
Want some more (y/n)?
n
Good night!
```

Table 2: Sound and Calories

Food item	Sound	Calories
Chips	Crunch Crunch	200
IceCream	Slurp Slurp	500
Pizza	Chomp Chomp	800

Solution Design: The class-diagram shows all classes and their members. Table 4 describes each class and its methods. Table 5 maps console I/O with various methods and their actions.

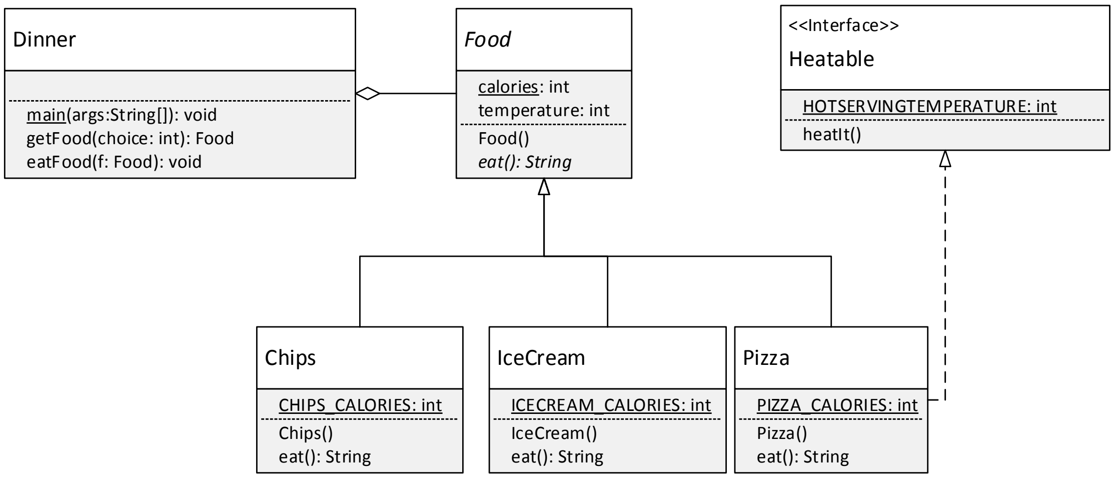

Figure 1: Class diagram

Instructions

- Download Dinner.java, Chips,java, and TestDinner.java from Canvas
- Create practice7 package and import the three java files into this package
- Create **Heatable** interface, and the other 3 classes (**Food**, **IceCream**, and **Pizza**) as required. Look at Chips.java code provided for hints. Pizza must implement Heatable interface. Then complete **Dinner**.java as directed. As you create these classes, keep testing your code using TestDinner.java .

Table 3: Class Descriptions

Class	Methods and descriptions
Food	• It has one class (static) variable: calories, and one member variable: temperature • Its constructor prints "Here comes food!" on the console. • Its eat() method is abstract
Heatable	• Has one constant named HOTSERVINGTEMPERATURE = 165° F. • Has one abstract method heatIt() that should set the food temperature to the desired temperature.
Chips	• Code provided
IceCream	• Extends Food class • Has constant ICECREAM_CALORIES as 500 • Its constructor increments 'calories' counter by ICECREAM_CALORIES and prints "Serving Ice cream" • Its eat method returns "Slurp Slurp" as a string value
Pizza	• Extends Food class and implements Heatable interface • Has constant PIZZA_CALORIES as 800 • The heatIt() method sets Pizza temperature to HOTSERVINGTEMPERATURE constant value. • Its constructor increments 'calories' counter by PIZZA_CALORIES and prints "Serving Pizza" • Its eat method returns "Chomp Chomp" as a string value
Dinner	• main(): presents menu-choices to the user, takes user input and passes it to getFood(). When getFood() returns the chosen food-item, it passes it as a parameter to the eatFood() method. Every time user chooses a food-item, it prints the updated message: "You have consumed X calories" on the console. Note that 'calories' is a static variable in Food class, as static variables are shown as underlined. • getFood(choice: int): based on the value of choice parameter, returns one of the 3 food-items. • eatFood(f: Food): If the food is Pizza, then it invokes its heatIt() method. For all items, It then invokes the eat() method of the food-item and prints the string returned by it

Table 4: Console I/O mapped to methods

Console I/O	Method and some description
`What would you like to eat?`	`main() method runs this in a loop`
`1. Pizza`	
`2. Chips`	
`3. Ice cream`	
`1`	*`User wants Pizza`*
`Here comes food!`	`This message printed from Food() constructor`
`Serving Pizza`	`This message printed from Pizza() constructor`
`Now its hot @ 165 degrees`	`This message printed from Pizza's heatIt() method`
`Chomp Chomp`	`This message printed from Pizza's eat() method`
`You have consumed 800 calories`	`Print the value of calorie from Food class`
`Want some more (y/n)?`	`If input is 'n' then break from loop, else continue back to displaying first menu.`
`y`	
`What would you like to eat?`	`main() method loop`
`1. Pizza`	
`2. Chips`	
`3. Ice cream`	
`2`	*`User chooses Chips`*
`Here comes food!`	`This message printed from Food() constructor`
`Serving Chips`	`This message printed from Chips() constructor`
`Crunch Crunch`	`This message from Chips' eat() method`
`You have consumed 1000 calories`	`Print the value of calorie from Food class`
`Want some more (y/n)?`	`If input is 'n' then break from loop, else continue back to displaying first menu.`
`y`	
`What would you like to eat?`	`main() method loop`
`1. Pizza`	
`2. Chips`	

`3. Ice cream`	
`3`	*`User wants Ice cream`*
`Here comes food!`	`This message printed from Food() constructor`
`Serving Icecream`	`This message printed from IceCream() constructor`
`Slurp Slurp`	`This message from IceCream’s eat() method`
`You have consumed 1500 calories`	`Print the value of calorie from Food class`
`Want some more (y/n)?`	`If input is ‘n’ then break from loop, else continue back to displaying first menu.`
`n`	
`Good night!`	`END`