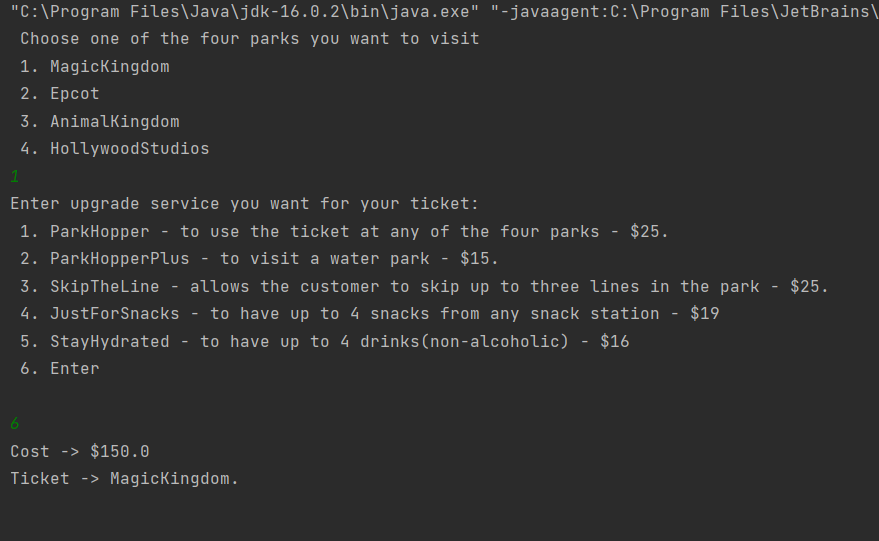
**CODE:**

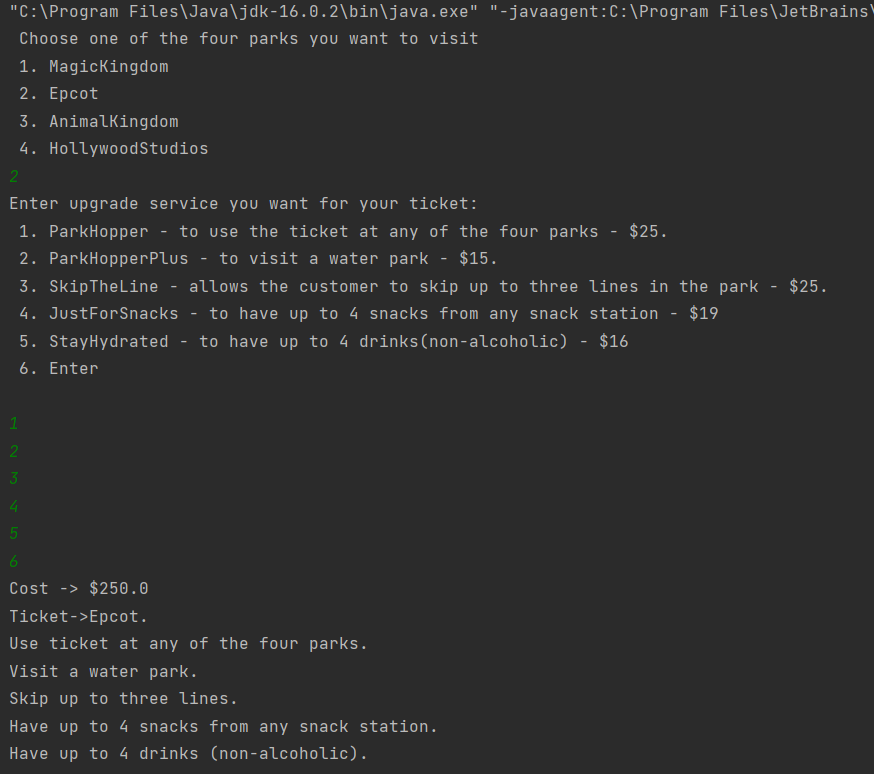
import java.util.Scanner;  
  
//abstract component - Ticket  
abstract class Ticket{  
 public abstract double cost();  
 public abstract String description();  
  
 @Override  
 public abstract String toString();  
}  
  
//concrete components - MagicKingdom, Epcot, AnimalKingdom, HollywoodStudios  
class MagicKingdom extends Ticket {  
  
 @Override  
 public double cost() {  
 return 150;  
 }  
  
 @Override  
 public String description() {  
 return "MagicKingdom.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket -> "+description();  
 }  
}  
  
class Epcot extends Ticket {  
  
 @Override  
 public double cost() {  
 return 150;  
 }  
  
 @Override  
 public String description() {  
 return "Epcot.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket -> "+description();  
 }  
}  
  
class AnimalKingdom extends Ticket {  
  
 @Override  
 public double cost() {  
 return 150;  
 }  
  
 @Override  
 public String description() {  
 return "AnimalKingdom.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket -> "+description();  
 }  
}  
  
class HollywoodStudios extends Ticket {  
  
 @Override  
 public double cost() {  
 return 150;  
 }  
  
 @Override  
 public String description() {  
 return "HollywoodStudios.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket -> "+description();  
 }  
}  
  
//abstract decorator - Upgrade  
abstract class Upgrade extends Ticket { }  
  
//concrete decorators - ParkHopper, ParkHopperPlus, SkipTheLine, JustForSnacks, StayHydrated  
class ParkHopper extends Upgrade{  
  
 Ticket ticket;  
 public ParkHopper(Ticket ticket){  
  
 this.ticket=ticket;  
 }  
  
 @Override  
 public double cost() {  
 return 25+ticket.cost();  
 }  
  
 @Override  
 public String description() {  
 return ticket.description()+"Use ticket at any of the four parks.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket->"+description();  
 }  
}  
  
class ParkHopperPlus extends Upgrade{  
 Ticket ticket;  
  
 public ParkHopperPlus(Ticket ticket) {  
  
 this.ticket=ticket;  
 }  
  
 @Override  
 public double cost() {  
 return 15+ticket.cost();  
 }  
  
 @Override  
 public String description() {  
 return ticket.description()+"Visit a water park.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket->"+description();  
 }  
}  
  
class SkipTheLine extends Upgrade{  
  
 Ticket ticket;  
  
 public SkipTheLine(Ticket ticket)  
 {  
  
 this.ticket=ticket;  
 }  
 @Override  
 public double cost() {  
 return 25+ticket.cost();  
 }  
  
 @Override  
 public String description() {  
 return ticket.description()+"Skip up to three lines.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket->"+description();  
 }  
}  
  
class JustForSnacks extends Upgrade{  
 Ticket ticket;  
  
 public JustForSnacks(Ticket ticket)  
 {  
  
 this.ticket=ticket;  
 }  
 @Override  
 public double cost() {  
 return 19+ ticket.cost();  
 }  
  
 @Override  
 public String description() {  
 return ticket.description()+"Have up to 4 snacks from any snack station.\n";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket->"+description();  
 }  
}  
  
class StayHydrated extends Upgrade{  
 Ticket ticket;  
  
 public StayHydrated(Ticket ticket)  
 {  
  
 this.ticket=ticket;  
 }  
 @Override  
 public double cost() {  
 return 16+ ticket.cost();  
 }  
  
 @Override  
 public String description() {  
 return ticket.description()+"Have up to 4 drinks (non-alcoholic).";  
 }  
  
 @Override  
 public String toString() {  
 return "Cost -> $"+cost()+"\nTicket->"+description();  
 }  
}  
  
  
  
  
public class Decorator {  
 public static void main(String[] args) {  
 Scanner sc=new Scanner(System.*in*);  
 Ticket ticket=null;  
 System.*out*.println(" Choose one of the four parks you want to visit");  
 System.*out*.println(" 1. MagicKingdom \n 2. Epcot \n 3. AnimalKingdom \n 4. HollywoodStudios");  
 int ch=sc.nextInt();  
 switch (ch) {  
 case 1 -> {ticket=new MagicKingdom();}  
 case 2 ->{ ticket = new Epcot();}  
 case 3 -> {ticket = new AnimalKingdom();}  
 case 4 -> {ticket = new HollywoodStudios();}  
 default -> {System.*out*.println("invalid choice");}  
 }  
  
 System.*out*.println("Enter upgrade service you want for your ticket:");  
 System.*out*.println("""  
 1. ParkHopper - to use the ticket at any of the four parks - $25.\s  
 2. ParkHopperPlus - to visit a water park - $15.\s  
 3. SkipTheLine - allows the customer to skip up to three lines in the park - $25.\s  
 4. JustForSnacks - to have up to 4 snacks from any snack station - $19\s  
 5. StayHydrated - to have up to 4 drinks(non-alcoholic) - $16\s  
 6. Enter""".indent(1));  
 boolean flag=true;  
 while (flag)  
 {  
  
 int choice= sc.nextInt();  
  
 switch (choice)  
 {  
 case 1 ->{  
 ticket=new ParkHopper(ticket);  
 }  
 case 2 ->{  
 ticket=new ParkHopperPlus(ticket);  
 }  
 case 3->{  
 ticket=new SkipTheLine(ticket);  
 }  
 case 4->{  
 ticket=new JustForSnacks(ticket);  
 }  
 case 5->{  
 ticket=new StayHydrated(ticket);  
 }  
 default -> {  
  
 flag=false;  
 }  
 }  
 }  
 System.*out*.println(ticket);  
  
  
 }  
  
}

**OUTPUT:**

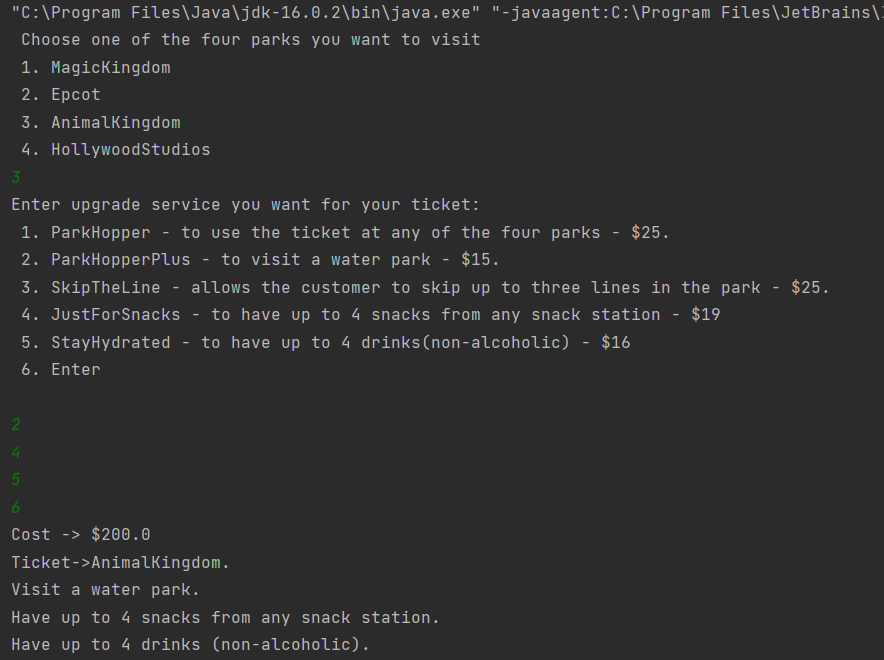
**1.**

****

**2.**

****

**3.**

****