CCPRGG2L
Intermediate Programming

Long Exam 2

By
Cruz, Tyrel P.

1.

PLAYER CLASS

```
// Filename: Player.java

import java.util.Scanner;

public class Player {
   String name;
   String team;
   int jerseyNumber;

   public void readPlayer() {
      //method body

      Scanner scan = new Scanner(System.in);
      System.out.print("Enter player name: ");
      name = scan.nextLine();
      System.out.print("Enter team name: ");
      team = scan.nextLine();
      System.out.print("Enter jersey number: ");
      jerseyNumber = scan.nextInt();
   }
}
```

COMPARE CLASS

```
// Filename: ComparePlayers.java

public class ComparePlayers extends Player {

   public static void main(String[] args) {
      Player player1 = new Player();
      Player player2 = new Player();

      //read player1 info
      System.out.println("----Player 1 Information----");
      player1.readPlayer();

      //read player2 info
      System.out.println("----Player 2 Information----");
      player2.readPlayer();
```

```
        //calls equal method
        if (equals(player1, player2)) {
            System.out.println("\n====Players are the same===");
        } else {
            System.out.println("\n====Players are different====");
        }
    }

    public static boolean equals(Player player1, Player player2) {

        return player1.team.equals(player2.team) && player1.jerseyNumber ==
player2.jerseyNumber;
    }
}
```

```
--- exec-maven-plugin:3.1.0:exec (default-cli) @ OOP ---
----Player 1 Information----
Enter player name: STLLR
Enter team name: BANDITS
Enter jersey number: 01
----Player 2 Information----
Enter player name: PRIMO
Enter team name: BANDITS
Enter jersey number: 02

====Players are different====
------------------------------------------------------------------------
BUILD SUCCESS
```

```
] --- exec-maven-plugin:3.1.0:exec (default-cli) @ OOP ---
 ----Player 1 Information----
 Enter player name: Tyrel
 Enter team name: Cruz
 Enter jersey number: 01
 ----Player 2 Information----
 Enter player name: Tyrel
 Enter team name: Cruz
 Enter jersey number: 01

 ====Players are the same===
 ------------------------------------------------------------------------
 BUILD SUCCESS
```

2.

HORSE CLASS

```
package two;

 public class Horse {
    private String name;
    private String color;
    private int birthYear;

    public Horse(String name, String color, int year) {
        setName(name);
        setColor(color);
        setBirthYear(year);
    }

    public void setName(String Name) {
        this.name = Name;
    }

    public void setColor(String Color) {
        this.color = Color;
    }

    public void setBirthYear(int Year) {
        this.birthYear = Year;
    }

    public String getName() {
        return name;
    }

    public String getColor() {
        return color;
    }

    public int getBirthYear() {
        return birthYear;
    }
}
```

RACE HORSE CLASS

```
package two;

public class RaceHorse extends Horse {
    private int nof;

    public RaceHorse(String name, String color, int birthYear, int nof) {
        super(name, color, birthYear);
        this.nof = nof;
    }

    public int getnof() {
        return nof;
    }

    public void setnof(int nof) {
        this.nof = nof;
    }
}
```

DEMO HORSES CLASS

```
package two;

public class DemoHorses {

    public static void main(String[] args) {
        Horse kalesa = new Horse("SKEPPY", "Yellow", 1990);
        RaceHorse karera = new RaceHorse("STALLION", "Black", 2003, 5);

        System.out.println("|| HORSE = KALESA | RACEHORSE = KARERANG KABAYO ||");

        System.out.println("\nKalesa"
                + "\nName: " + kalesa.getName()
                + ", Color: " + kalesa.getColor()
                + ", BirthYear: " + kalesa.getBirthYear());

        System.out.println("\nKarerang Kabayo"
                + "\nName: " + karera.getName()
```

```
            + ", Color: " + karera.getColor()
            + ", BirthYear: " + karera.getBirthYear()
            + ", Number of Races: " + karera.getnof());


    }
}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Finals ---
|| HORSE = KALESA | RACEHORSE = KARERANG KABAYO ||

Kalesa
Name: SKEPPY, Color: Yellow, BirthYear: 1990

Karerang Kabayo
Name: STALLION, Color: Black, BirthYear: 2003, Number of Races: 5
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

CANDLE CLASS

```
package three;

public class Candle {
    public String color;
    public double height;
    public double price;


    public String getColor() {
        return color;
    }

    public double getHeight() {
        return height;
    }

    public double getPrice() {
        return price;
    }

    public void setColor(String c) {
        this.color = c;
    }

    public void setHeight(double h) {
        this.height = h;
        price = height * 2;
    }
}
```

SCENTED CANDLE CLASS

```
package three;

public class ScentedCandle extends Candle {
```

```
    public String scent;

    public String getScent(){
        return scent;
    }

    public void setScent(String s){
        scent=s;
    }

    @Override
    public void setHeight(double h){
        super.height=h;
        super.price=super.height*3;
       }
}
```

DEMO CANDLES

```
package three;
import java.util.Scanner;

public class DemoCandles {

    public DemoCandles() {
    }

    public static void main(String args[])
    {
        Scanner scanner = new Scanner(System.in);

        Candle  c= new Candle();
        System.out.println(" || Normal Candle Color, Height and its Price per Inch || ");
        System.out.print("Enter Candle Color: ");
        String color = scanner.next();
        c.setColor(color);
        System.out.print("Enter Candle Height: ");
        double height = scanner.nextDouble();
        c.setHeight(height);

System.out.println("_____
_____");
```

```
        System.out.println("\n Candle Color: "+c.getColor()+"\n Candle Height: "+c.getHeight()+"\n
Candle Price: "+c.getPrice());

        System.out.println("");

        ScentedCandle sc=new ScentedCandle();
        System.out.println(" || Scented Candle Color, Height, Scent and its Price per Inch || ");
        System.out.print("Enter Scented Candle Color: ");
        String scentedcolor = scanner.next();
        sc.setColor(scentedcolor);
        System.out.print("Enter Scented Candle Height: ");
        double scentedCandleHeight = scanner.nextDouble();
        sc.setHeight(scentedCandleHeight);
        System.out.print("Enter Scent: ");
        String scent = scanner.next();
        sc.setScent(scent);

System.out.println("_____
_____");
        System.out.println("\n Scented Candle Height: "+sc.getHeight()+"\n Scent:
"+sc.getScent()+"\n ScentedCandle Price: "+sc.getPrice());
    }
}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Finals ---
 || Normal Candle Color, Height and its Price per Inch ||
Enter Candle Color: White
Enter Candle Height: 5
_____

 Candle Color: White
 Candle Height: 5.0
 Candle Price: 10.0

 || Scented Candle Color, Height, Scent and its Price per Inch ||
Enter Scented Candle Color: Blue
Enter Scented Candle Height: 5
Enter Scent: Almond
_____

 Scented Candle Height: 5.0
 Scent: Almond
 ScentedCandle Price: 15.0
------------------------------------------------------------------------
BUILD SUCCESS
```

4. A

BOOK CLASS

```
package four;

public abstract class Book {
    private String title;
    double price;

    public Book(String title){
        super();
        this.title = title;
    }

    public String getbookTitle() {
        return title;
    }

    public double getbookPrice() {
        return price;
    }

    public abstract void setPrice();

}
```

FICTION CLASS

```
package four;

public class Fiction extends Book {
    public Fiction(String title) {
        super(title);
        setPrice();
    }

    @Override
    public void setPrice() {
```

```
        price = 249.90;
    }
}
```

NON FICTION CLASS

```
package four;

public class NonFiction extends Book {
    public NonFiction(String title) {
        super(title);
        setPrice();
    }

    @Override
    public void setPrice() {
        super.price = 379.95;
    }
}
```

USE BOOK CLASS

```
package four;

public class UseBook {
    public static void main(String[] args) {
        Fiction f = new Fiction("Noli Me Tangere");
        System.out.println(f.getbookTitle() + " - " + f.getbookPrice());

        NonFiction nonF = new NonFiction("Filipino popular Tales");
        System.out.println(nonF.getbookTitle() + " - " + nonF.getbookPrice());
    }
}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Finals ---
Noli Me Tangere - 249.9
Filipino popular Tales - 379.95
------------------------------------------------------------------------
BUILD SUCCESS
```

4. B

```
package four;

public class BookArray {
   public static void main(String[] args) {

      Book bookArray[]={
         new Fiction("Harry Potter"),
         new Fiction("El Filibusterismo"),
         new Fiction("Noli Me Tangere"),
         new Fiction("Ilustrado"),
         new Fiction("Si Janus Silang at ang Tiyanak ng Tabon"),

         new NonFiction("\nThe Philippine Revolution"),
         new NonFiction("The Conjugal Dictatorship of Ferdinand and Imelda Marcos"),
         new NonFiction("Pilgrim in America"),
         new NonFiction("Without Seeing the Dawn"),
         new NonFiction("Reportage on Politics")
         };

      for (int i = 0; i < bookArray.length; i++) {
         System.out.println(bookArray[i].getbookTitle() + " - " + bookArray[i].getbookPrice());
      }
   }
}
```

```
Harry Potter - 249.9
El Filibusterismo - 249.9
Noli Me Tangere - 249.9
Ilustrado - 249.9
Si Janus Silang at ang Tiyanak ng Tabon - 249.9

The Philippine Revolution - 379.95
The Conjugal Dictatorship of Ferdinand and Imelda Marcos - 379.95
Pilgrim in America - 379.95
Without Seeing the Dawn - 379.95
Reportage on Politics - 379.95
------------------------------------------------------------------------
BUILD SUCCESS
```

5. A

PHONE CALL CLASS

```
abstract class PhoneCall {
String phoneNumber;
double price;

    public PhoneCall(String num) {
        this.phoneNumber=num;
        this.price=0.0;
    }


    public void setPrice(double pr) {
        price=pr;
    }

    public String getPhoneNumber(){
    return phoneNumber;
    }

    public double getPrice(){
    return price;
    }

    public abstract void getInfo();

}
```

INCOMING PHONE CALL CLASS

```
class IncomingPhoneCall extends PhoneCall {

    public IncomingPhoneCall(String num) {
        super(num);
        setPrice();
    }

    public void setPrice(){
```

```
        price = 0.02;
    }

    @Override
    public void getInfo() {
        System.out.println("\nNumber : " + getPhoneNumber());
        System.out.println("Price : " + getPrice());
    }
}
```

OUTGOING CLASS

```
class OutgoingPhoneCall extends PhoneCall {

    int minutes;

    public OutgoingPhoneCall(String num, int mins) {
        super(num);
        minutes=mins;
        setPrice(4.00);
    }

    @Override
    public void getInfo() {
        System.out.println("\nNumber : "+ getPhoneNumber());
        System.out.println("Rate per Minute : "+ getPrice());
        System.out.println("Total minutes : "+ minutes);
        System.out.println("Total Price : "+ (minutes*price));
    }

    @Override
    public String getPhoneNumber() {
        return phoneNumber;
    }



    @Override
    public double getPrice() {
        return price;
    }
}
```

DEMOPHONECALLS CLASS

```
public class DemoPhoneCalls {

    public static void main(String[] args) {

        IncomingPhoneCall inCall = new IncomingPhoneCall("212-555-3096");
        OutgoingPhoneCall outCall = new OutgoingPhoneCall("312-874-0232", 10);
        inCall.getInfo();
        outCall.getInfo();
}

}
```

```
Number : 212-555-3096
Price : 0.02

Number : 312-874-0232
Rate per Minute : 4.0
Total minutes : 10
Total Price : 40.0
------------------------------------------------------------------------
BUILD SUCCESS
```

5. B

PHONE CALL ARRAY CLASS

```
public class PhoneCallArray

{

    public static void main(String[] args)

            {
```

```
        PhoneCall randcall[] = new PhoneCall[10];

    int x;

    randcall[0] = new IncomingPhoneCall("123-895-7542");
    randcall[1] = new OutgoingPhoneCall("254-986-5324",25);
    randcall[2] = new IncomingPhoneCall("157-789-5421");
    randcall[3] = new OutgoingPhoneCall("154-853-1235",90);
    randcall[4] = new IncomingPhoneCall("125-894-6584");
    randcall[5] = new OutgoingPhoneCall("234-859-6548",250);
    randcall[6] = new IncomingPhoneCall("236-986-5214");
    randcall[7] = new IncomingPhoneCall("546-235-6987");
    randcall[8] = new IncomingPhoneCall("231-856-9854");
    randcall[9] = new IncomingPhoneCall("946-589-7985");



     for(x = 0; x < randcall.length; ++x)

        System.out.println("\nPhone Number: " + randcall[x].getPhoneNumber() + " \nCosts: "
+ randcall[x].getPrice());

        }

}
```

```
Phone Number: 123-895-7542
Costs: 0.02

Phone Number: 254-986-5324
Costs: 4.0

Phone Number: 157-789-5421
Costs: 0.02

Phone Number: 154-853-1235
Costs: 4.0

Phone Number: 125-894-6584
Costs: 0.02

Phone Number: 234-859-6548
Costs: 4.0

Phone Number: 236-986-5214
Costs: 0.02

Phone Number: 546-235-6987
Costs: 0.02

Phone Number: 231-856-9854
Costs: 0.02

Phone Number: 946-589-7985
Costs: 0.02
------------------------------------------------------------------------
BUILD SUCCESS
```

6.

TURNER CLASS

```
package six;


interface Turner {
    void turn();
}
```

LEAF CLASS

```
package six;

class Leaf implements Turner {
    @Override
    public void turn() {
        System.out.println("Changing colors");
    }
}
```

PAGE CLASS

```
package six;

class Page implements Turner {
    @Override
    public void turn() {
        System.out.println("Going to the next page");
    }
}
```

PANCAKE CLASS

```
package six;
```

```
class Pancake implements Turner {
    @Override
    public void turn() {
        System.out.println("Flipping");
    }
}
```

DEMOTURNERS

```
package six;

public class DemoTurners {
    public static void main(String[] args) {
        // Leaf
        Turner t = new Leaf();
        t.turn();

        // Page
        t = new Page();
        t.turn();

        // Pancake
        t = new Pancake();
        t.turn();
    }
}
```

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Finals ---
Changing colors
Going to the next page
Flipping
------------------------------------------------------------------------
BUILD SUCCESS
```

7.

RUNNER CLASS

```
package seven;

public interface Runner {

public abstract void run();

}
```

MACHINE CLASS

```
package seven;

public class Machine implements Runner {

@Override
public void run() {
System.out.println("----MACHINE----");
System.out.println("(Machine that runs) is a general term used to describe any type of device or system that is capable of performing a set of operations or functions. \nIn the context of computers or other electronic devices, a machine that runs typically refers to a system that is powered on and actively executing code or applications. \nThis could include a desktop computer, laptop, tablet, smartphone, server, or other similar devices that are designed to perform specific tasks or provide certain functionality. \nOverall, the term (machine that runs) simply refers to any type of technology or device that is currently operational and performing its intended functions.");

        }
}
```

ATHLETE CLASS

```
package seven;

public class Athlete implements Runner {
```

```
@Override
public void run() {
System.out.println("");
System.out.println("----ATHLETE----");
System.out.println("\"Athlete running\" generally refers to a person who is engaged in the physical activity of running as a sport or form of exercise. \nThe term \"athlete\" typically implies that the person is trained, skilled, and/or competitive in their running ability, and may participate in various types of running events such as track and field, marathons, cross-country races, or other similar competitions. \nRunning as an athletic activity requires a significant amount of physical endurance, strength, and technique, and may involve training programs, specialized equipment, and a focus on proper form and nutrition. \nOverall, the term \"athlete running\" is commonly used to describe a person who is actively participating in the sport of running at a competitive or skilled level.");

        }
}
```

POLITICAL CANDIDATE

```
package seven;

public class PoliticalCandidate implements Runner {

@Override
public void run() {

System.out.println("");
   System.out.println("----POLITICAL CANDIDATE----");
System.out.println("\"Political candidate\" refers to a person who seeks elected office within a political system. This can include a variety of positions such as president, senator, governor, mayor, or other similar roles. \nPolitical candidates typically campaign and run for office by promoting their qualifications, views, and policies to the public, and may participate in debates, rallies, interviews, or other forms of communication in order to build support and gain votes. \nPolitical candidates may belong to a specific political party or run as an independent, and may be subject to various laws and regulations regarding fundraising, advertising, and other aspects of the campaign process. \nOverall, the term \"political candidate\" refers to an individual who is actively seeking to be elected to a political office within a given political system.");

        }
}
```

DEMO RUNNERS CLASS

```
package seven;

public class DemoRunners {

public static void main(String[] args) {

Machine r1 = new Machine();

Athlete r2 = new Athlete();

PoliticalCandidate r3 = new PoliticalCandidate();

r1.run();

r2.run();

r3.run();

        }

}
```

```
----MACHINE----
(Machine that runs) is a general term used to describe any type of device or system that is capable of performing a set of operations or functions.
In the context of computers or other electronic devices, a machine that runs typically refers to a system that is powered on and actively executing code or
This could include a desktop computer, laptop, tablet, smartphone, server, or other similar devices that are designed to perform specific tasks or provide c
Overall, the term (machine that runs) simply refers to any type of technology or device that is currently operational and performing its intended functions.

----ATHLETE----
"Athlete running" generally refers to a person who is engaged in the physical activity of running as a sport or form of exercise.
The term "athlete" typically implies that the person is trained, skilled, and/or competitive in their running ability, and may participate in various types
Running as an athletic activity requires a significant amount of physical endurance, strength, and technique, and may involve training programs, specializec
Overall, the term "athlete running" is commonly used to describe a person who is actively participating in the sport of running at a competitive or skilled

----POLITICAL CANDIDATE----
"Political candidate" refers to a person who seeks elected office within a political system. This can include a variety of positions such as president, sena
Political candidates typically campaign and run for office by promoting their qualifications, views, and policies to the public, and may participate in deba
Political candidates may belong to a specific political party or run as an independent, and may be subject to various laws and regulations regarding fundrai
Overall, the term "political candidate" refers to an individual who is actively seeking to be elected to a political office within a given political system.
------------------------------------------------------------------------
BUILD SUCCESS
```