INFO1113 Object-Oriented Programming

Week 12B: Revision Part 2

Copyright Warning

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Examination Topics

- Simple class inheritance
- Interfaces and abstract classes
- UML Class Hierarchy Diagrams
- Instance and static variables
- Collections and Enums
- Recursion
- Wildcards
- Generics and Type Bounds
- Overloading and Overriding
- Testing

Review Material

```
/** An enumeration of card suits. */
enum Suit
{
    CLUBS("black"), DIAMONDS("red"), HEARTS("red"),
    SPADES("black");

    private final String color;

    private Suit(String suitColor)
    {
        color = suitColor;
    }
    public String getColor()
    {
        return color;
    }
}
```

If cardSuit is an instance of Suit and is assigned the value Suit.SPADES, what is returned by each of the following expressions?

a. System.out.println(cardSuit.ordinal())

3

Review Material

```
/** An enumeration of card suits. */
enum Suit
{
    CLUBS("black"), DIAMONDS("red"), HEARTS("red"),
    SPADES("black");

    private final String color;

    private Suit(String suitColor)
    {
        color = suitColor;
    }
    public String getColor()
    {
        return color;
    }
}
```

If cardSuit is an instance of Suit and is assigned the value Suit.SPADES, what is returned by each of the following expressions?

a. System.out.println(cardSuit.ordinal()) 3

b. System.out.println(cardSuit.equals(Suit.CLUBS)) false

Review Material

```
/** An enumeration of card suits. */
enum Suit
{
    CLUBS("black"), DIAMONDS("red"), HEARTS("red"),
    SPADES("black");

    private final String color;

    private Suit(String suitColor)
    {
        color = suitColor;
    }
    public String getColor()
    {
        return color;
    }
}
```

If cardSuit is an instance of Suit and is assigned the value Suit.SPADES, what is returned by each of the following expressions?

a. System.out.println(cardSuit.ordinal()) 3

b. System.out.println(cardSuit.equals(Suit.CLUBS)) false

c. System.out.println(cardSuit.getColor()) black

Review Material

```
/** An enumeration of card suits. */
enum Suit
{
    CLUBS("black"), DIAMONDS("red"), HEARTS("red"),
    SPADES("black");

    private final String color;

    private Suit(String suitColor)
    {
        color = suitColor;
    }
    public String getColor()
    {
        return color;
    }
}
```

If cardSuit is an instance of Suit and is assigned the value Suit.SPADES, what is returned by each of the following expressions?

a. System.out.println(cardSuit.ordinal())	3
b. System.out.println(cardSuit.equals(Suit.CLUBS))	false
c. System.out.println(cardSuit.getColor())	black
d. System.out.println(cardSuit)	SPADES

Review Material

Write a program that allows user to enter the marks of the students using standard I/O. Store all the marks in an ArrayList and print the average mark of the students.

The inputted mark must be between 0 and 100. In case, the mark is out of bound, throw a InvalidMarkException. When a InvalidMarkException is thrown, the getMessage() should return “Invalid Mark.” You need to have appropriate catch block for any other exceptions.

Review Material

```
class InvalidMarkException extends Exception{

   public InvalidMarkException(){

      super("Invalid Mark.");

   }

}
```

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

      ArrayList<Integer> marks = new ArrayList<>();
      Scanner scan = new Scanner(System.in);
      int SumOfMarks = 0;

      try{
         while(scan.hasNext()){
            int mark = scan.nextInt();
            if(mark < 0 || mark > 100)
               throw new InvalidMarkException();
            else {
               marks.add(mark);
               SumOfMarks += mark;
            }
         }

         System.out.println(SumOfMarks/marks.size());

      }catch(InvalidMarkException e){
         System.out.println(e.getMessage());
      }catch(Exception e){
         e.printStackTrace();
      }
   }
}
```

Review Material

Find the errors. Specify the line numbers and mention the corrections required

```
1.  public class PrintMax {
2.      public static void main(String args) {
3.          int a = args.size();
4.          int max = 0;
5.          for (int i; i < a; ++i) {
6.              if (args[i] > max)
7.                  max = args[i]);
8.              else
9.                  max = max;
10.         }
11.         System.out.println(max);
12.     }
13. }
```

Review Material

Find the errors. Specify the line numbers and mention the corrections required

```
1.  public class PrintMax {
2.      public static void main(String args) {   →   public static void main(String[] args) {
3.          int a = args.size();                 →   int a = args.length;
4.          int max = 0;
5.          for (int i; i < a; ++i) {            →   for (int i = 0; i < a; ++i) {
6.              if (args[i] > max)               →   if (Integer.parseInt(args[i]) > max)
7.                  max = args[i]);              →   max = Integer.parseInt(args[i]);
8.              else
9.                  max = max;
10.         }
11.         System.out.println(max);
12.     }
13. }
```

Review Material

Create an abstract class *PayCalculator* that has an attribute payRate given in dollars per hour. The class should have an abstract method computePay(hours) to return the pay for a given amount of time.

Derive a class *RegularPay* from *PayCalculator*. It should have a constructor that has a parameter for the payRate. It should implement the computePay(hours) method which returns the pay for a given amount of time.

Review Material

Create an abstract class *PayCalculator* that has an attribute payRate given in dollars per hour. The class should have an abstract method computePay(hours) to return the pay for a given amount of time.

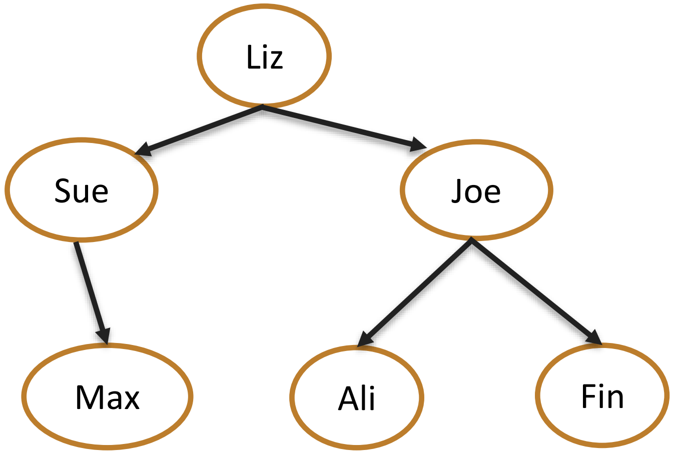
Derive a class *RegularPay* from *PayCalculator*. It should have a constructor that has a parameter for the payRate. It should implement the computePay(hours) method which returns the pay for a given amount of time.

```
abstract class PayCalculator{
    double payRate;
    public abstract double computePay(double hours);
}

class RegularPay extends PayCalculator{
    public RegularPay(double payRate){
        this.payRate = payRate;
    }
    public double computePay(double hours){
        return hours * payRate;
    }
}
```

Review Material

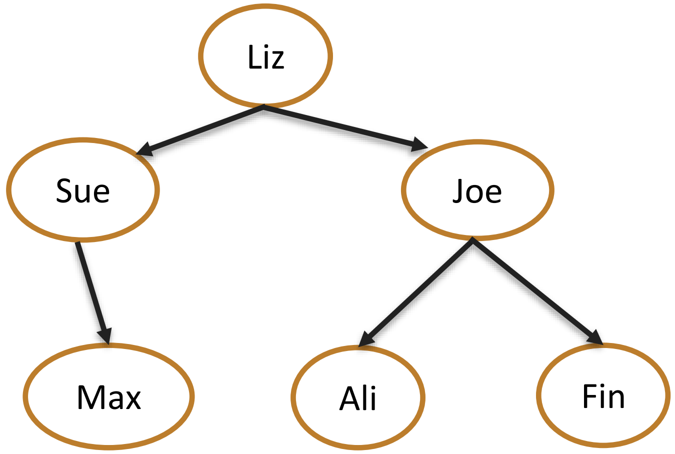
Consider the following FamilyMember class. Write a recursive procedure to count the number of leaf nodes (family members with no children) in the family tree.

```
class FamilyMember {

    String name;
    List<FamilyMember> children;

    public FamilyMember(String name) {
        this.name = name;
        children = new ArrayList<>();
    }

    public void addChildren(FamilyMember f){
        children.add(f);
    }

    public int countLeaf() {
        //your implementation here
    }
}
```



Review Material

```
class FamilyMember {
    String name;
    List<FamilyMember> children;

    public FamilyMember(String name) {
        this.name = name;
        children = new ArrayList<>();
    }
    public void addChildren(FamilyMember f){
        children.add(f);
    }
    public int countLeaf() {
        int counter = 0;
        if(this.children.size() == 0)
            return 1;
        else{
            for(FamilyMember f : children)
                counter += f.countLeaf();
        }
        return counter;
    }
}
```



… and that’s it.

In future

Use the skills to make something awesome or learn something new!

- Learn C, C++, Javascript, C#, Something else
- Write a web application
- Write android applications
- Learn the new features in java 11
- … possibilities are endless, use the time you have now!

Teaching Team

Thanks to the teaching team of this semester!

- Andrew Esteban (TA)
- Madeleine Wagner
- Nejc Moskon
- Nikola Grujic
- Samuel Lin
- Sudeshna Sengupta
- Adam Ghanem
- Daniel Friedrich
- Finnegan Waugh
- Samarth Sehgal
- Zhiye Hong
- Ben Gane
- Sheikh Mohammad Mostakim Fattah

Unit of study survey

You have access to the unit of study survey

https://student-surveys.sydney.edu.au/students/

Please respond to this survey as we are interested in what we can improve with this unit.

I wish you all success in your life!

Thank you