REG NO:P101/0847G/18

UNIT TITTLE:OBJECT ORIENTED PROGRAMMING

UNIT CODE:COM 328

CAT ONE

1. A message must contain the number to call, the number of the sender, and the message text.

Define a class Message, with:

1. A method that creates an instance starting from an object of the class MessageText and two strings that represent the number to call and the number of the sender.

class Message{

void MessageText{

private String number\_call;

private String number\_sender;

}

}

1. A method that takes as parameter a phone number, and uses it to update the number to call of the message. [3 Marks]

//method with phone\_number as parameter

public String getCallNumber (String phone\_number) {

//update the number to call with the parameter phone\_number

this.number\_call = phone\_number;

return this.number\_call;

}

1. A method that prints the message. [2 Marks]

// method to print the message

public void printMessage(String newCallNumber) {

//calling the method with message and then print it

System.out.println(MessageText());

}

1. A positive integer is said to be prime if it is divisible only by 1 and by itself. Write a method that takes as parameter a positive integer, and returns a boolean that indicates whether the integer is prime or not. [5 Marks].

import java.util.Scanner;

class CheckPrime

{

public static void main(String args[])

{

int temp;

boolean IsPrime=true;

Scanner scan= new Scanner(System.in);

System.out.println("Enter a number:");

int num=scan.nextInt();

scan.close();

for(int i=2;i<=num/2;i++)

{

temp=num%i;

if(temp==0)

{

IsPrime=false;

break;

}

}

//If IsPrime is true then the number is prime else not

if(IsPrime)

System.out.println(num + " is a Prime Number");

else

System.out.println(num + " is not a Prime Number");

}

}

1. All methods in Java are dynamic polymorphic as a design choice. Discuss the advantages and disadvantages of making dynamic polymorphism optional for Java methods. [6 Marks]

Advantages of Dynamic Polymorphism

* It also allows subclasses to add its specific methods subclasses to define the specific implementation of same.
* Allows Java to support overriding of methods which is central for run-time polymorphism.
* It allows a class to specify methods that will be common to all of its derivatives while allowing subclasses to define the specific implementation of some or all of those methods.

Disdvantages

One cannot override static methods.

* One cannot override the private methods of a parent class.
* Final methods cannot be overridden.

1. In the try … catch … finally block, explain the type of code expected in each block. Use their syntax to show the answer.

try {

try-block

}

catch(ExceptionClass1 e) {

catch-block

}

catch(ExceptionClass2 e) {

catch-block

}

...

finally {

finally-block

}

• try-block : sequence of statements that will be executed under the control of the following catch clauses

• catch-block : sequence of statements that will be executed if a statement in the try-block generates

an exception of the type specified in the corresponding catch clause

• finally-block : sequence of statements that will be always executed (both in the case where the

try-block is executed without exceptions, and in the case where a catch-block is executed to catch an

exception).

1. Explain what is Exception propagation? Illustrate your answer with Java code snippet using throw new ArgumentException(“Illegal input to the method.”). [5 Marks]

* exception propagation occurs when an exception thrown from the top of the stack. When it is not caught, the exception drops down the call stack of the preceding method. If it is not caught there, it further drops down to the previous method. This continues until the method reaches the bottom of the call stack or is caught somewhere in between.

throw new exception\_class("Error message shown for the method that is illegal ");

CAT TWO

1. Define a class Person with the private data members: first name, last name, and date of birth; public methods: get name and get date of birth and a constructor to initialize the data members. [5 Marks]

class Person {

private String first\_name;

private String last\_name;

private String date\_of\_birth;

Person(String f\_name, String l\_name , String DOB) {

first\_name = f\_name;

last\_name = l\_name;

date\_of\_birth =DOB;

}

public String get\_name() {

return name;

}

public String get\_ date\_of\_birth () {

return date\_of\_birth;

}

}

1. Define another class Student that inherits from class Person and has the following properties: student registration number, address, and programmme and the public methods: get grade and a constructor to initialize student address and programme. Ensure that the Person constructor is called in the Student constructor. [6 Marks]

class Student extends Person{

int reg\_no;

String address;

String programme;

Student(int regno, String Address, String prog){

reg\_no = regno;

address = Address;

programme = prog;

Person();

}

public String get\_grade() {

return grade;

}

}

1. Using a scenario of choice (Such as Course Registration) Write a program that demonstrates inheritance, method Overriding and Overloading and polymorphism. Make sure to comment your program appropriately. [10 Marks]

//Creating a parent class.

Class vehicle{

float weight;

long price;

//method to demonstrate polymorphism

void get\_model(){

System.out.println(“…..”);

}

//constructor

Vehicle(float weight, long price){

weight = weight;

price = price;

}

  void display(float weight){

System.out.println(weight);

}

   void display(float weight, long price){

System.out.println(price / weight);

}

    void run(){

System.out.println("Vehicle is running");

}

}

class Honda extends Shape{

void draw(){

System.out.println("The model type is honda");

}

}

class Toyota extends Shape{

void draw(){System.out.println("The model type is toyota");}

}

// child Inheriting from parent class

class Bike extends Vehicle{

  //method overriding

  void run(){

System.out.println("Bike is running safely");

}  }

class DisplayResults{

public static void main(String args[]){

//Test overriding

Bike obj = new Bike();// creating object

  obj.run() ;// calling method to override run() in the parent class

//Test overloading

Vehicle.display(12.98);

Vehicle.display(12.98, 570000);

//Test polymorphism

Vehicle V;

V =new Honda();

V.Display();

V =new Honda();

V.Display();

}