1.a

Java: Java is a popular programming language based on oop (Object Oriented programming) which is used to develop mobile application, website etc.

Java is a Platform independence programming language. It can run on platform. There no need of rewrite java program while it was written in a OS and then run to another OS. We can easily run and compile the program on MacOS which was originally written on Windows os.

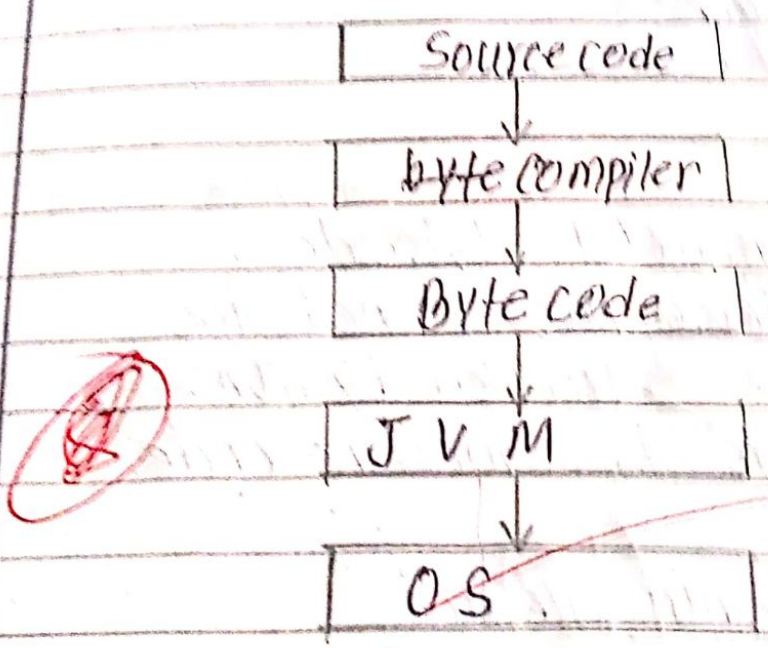


fig: Java compilation process

1a

The compilation process is as shown in figure:

i) At first a programmar write a java program in a notepad, jupiternotebook, intellij etc. The program has .java file.

ii) After source code written compiler compiles it. For compile we use command prompt for program written in notebook. find the location of program and enter cmd. Compiler changes source code to byte code which has .class file.
typing: javac programname.java
java Program
it is compiled.

iii) After bytecode generated bytecode verfier verifies the java has or not the specification then it is accepted by JVM. which changes bytecode to machine code.

iv) After that we get our output.

1b

We need coding standard to keep similar coding style all over the world. Coding standard helps to understand the code everyone all over the world. There is misunderstanding about the program if a programmer doesn't follow coding standard. A good programmer must follow the standard.

→ Naming standard rules:

SN	parameter	rules	Example
1	class	class keyword must be lowercase. The first letter of every word of a class must be capital and others are lowercase.	class HelloWorld { }
2	variable	variable must follow with datatype and variablename. datatype is lowercase except String's S and variablename is lowercase It must follows semicolon. If a variable has more than a word then after one word every word's first letter is capital.	int st String name; int roll; eg: String nameOfStudent;

15

Naming rules:

SN	Parameter/ method	Rules	Example
3	method	method must have inside a class. they follows with return time method name() and then braces. The name of function is in camelcase. If there is one word it is lowercase. If there is more than one word every word after firstword follows capital letter by first letter.	public class Nast { void computeSum() { } }
4	Package	It has package keyword and packagename. packagename. Package name is in lowercase.	eg: package file;

2a

Class: Class is the collection of similar types of objects. It is acts as container in Java. Eg: Student, Bird etc

Object: Object is the instance of a class. It is real world entity. A class can has many objects. Example. Ram, Sam etc are objects of class Student.

Constructor: Constructor are the member of a class which has not return type similar name as the class. It is used for data initialization and created when instance of a class is generated.

example:

```
public class Calculation {

public Calculation ( ) {
    int a = 4;
    int b = 5;
    System.out.println ("sum = " + (a+b));
}
public Calculation (int a, int b)
{
    this.a = a;
```

code 2a

```
    this.b = b;
    System.out.println (" sum of parameterized constructor
                        + (a+b) );
}
public static void main ( String... args )
{
    Calculation c = new Calculation ( );
    Calculation c1 = new Calculation ( 10,20);
}
}
```

2b

Package: Package is the collection of similar types of classes and interface.

Interface: Interface is a platform which provides multiple inheritance.

Syntax: interface interfacename {
}

eg: interface Parent {
}

~~Code:~~ creating package steps:

i) write keyword 'package' at first then write packagename and terminate with semicolon.

ii) After that define class and inside class follows method, variables etc.

iii) We can define multiple classes inside a package.

2b

code example :

```
package example;
public class Detail {
    void getInfo (String name, int roll) {
        this.name = name;
        this.roll = roll;
    }
    void display ( ) {
System.out.println ("Detail of students is:");
System.out.println ("Name:" + name);
System.out.println ("Rollno:" + roll);
}
public static void main (String... args) {
Detail d = new Detail ( );
d.getInfo ("Dinesh", 2);
d.display ( );
}
}
```

5

3a

Need of exception handling:

- to handle exception arises during program run.
- as exception are runtime error it doesn't give output, so to get output we must handle exception.

Code: Handle user defined exception:

```
public class GrettingLowBalance_Exception extends
        Exception {

public GrettingLowBalanceException (String str)
{
    super (str);
}
}

public class LowBalanceDemo {
    float balance = 50.26F;
    ~~while (balance LowBalanceDemo ( ))~~
{
  while (balance != 0) {
    if (balance < 5) {
```

9a

```
try {
    throw new CreditingLowBalanceException ("Please
                                    recharge on time");
}
catch ( CreditingLowBalanceException glbe) {
    System.out.println (glbe);
}
}
}
}
```

3b

Use of conditional statements

→ It is used to check the condition if it is true output gives else terminates. There is no output.

→ if we have need only one condition among various it is very suitable.

→ if, if else, if else if, switch, forloop, while loop, do while loop etc are the conditional statements.

code example : To find no. pass students among 48 students in Java.

```
import java.util.Scanner;
public class Result {
        int count = 0;
    Result ( )
{
    Scanner s = new Scanner (System.in);
     for ( int i = 1; i <= 48; i++ )
{
    System.out.println (" enter marks of student" +i);
     float marks[i] = new s.nextFloat();
}
```

3b

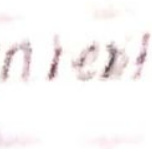
```
for (i = 1 ; i <= 48; i++)
{
  if (marks[i] > 45)
      count++ ;
{ System.out.print
}
}   void display ( ) {
  System.out.println (" pass students are = " + count);
}
}
public static void main ( String... args) {
  Result r = new Result ( );
     r. display ( );
}
}
```

4

Inheritance: It is feature of OOPs in which child class inherits the property of parent class.

Types of inheritance

a) Simple inheritance



→ As shown in figure simple inheritance has only one child of a parent.

code:

```
public class Parent {
    void show() {
    System.out.println("This is parent");
    }
}

public class Child extends Parent {
  void display() {

  System.out.println("This is child");
}
  public static void main(String args) {
     Child c = new Child();
      c.show();
```

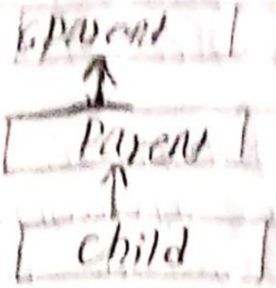
11. Multiple In

pa.

as display ( ) ;
}
}

b) Multilevel Inheritance

→ One It has level of inheritance as shown in figure.



code:

```
public class GrandParent {
void GrandParent ( ) {
System.out.println ("I'm grand parent");
}
}
public class Parent extends GrandParent {
void Parent ( ) {
System.out.println ("I'm parent");
}
}
public class Child extends Parent {
void child ( ) {
```

National Academy of science and Technology
[Affiliated to Pokhara University]
Dhangadhi, Kailali (Nepal)
Accredited by University Grants Commission (UGC), Nepal (2022)

3521

Additional answerbook No. ..... Invigilator's Signature .....
Number of additional answerbooks used : .....

Name of the student : ..... Programme .....
Roll No (Class) : ..... Exam : ..... Grade/Semester ..... Exam .....
Paper ..... Date .....

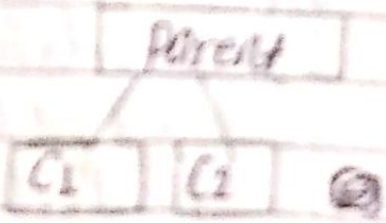
* Students are required to write answer on both sides starting from the first page of this copy.
* This answerbook must be attached properly with the original answer book before delivering to the invigilator.

5a

```
System.out.println(" I'm a child");
}
public static void main (String... args) {
Child c = new Child ();
c.grandParent ();
c.Parent ();
c.Child ();
}
}
```

c) Hierarchical Inheritance

→ One parent has multiple children.

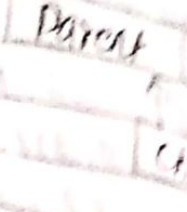


```
class Parent {
    void Parent() {
    System.out.println("Parent");
    }
}

Public class C1 extends Parent {
    void C1() {
    System.out.println("child");
    }
}

public class C2 extends Parent {
    void C2() {
    System.out.println("Child 2");
    }
}

public class HierarchicalDemo {
public static void main() {
    Parent p = new Parent();
    C1 c1 = new C1();
    C2 c2 = new C2();
        p.Parent();
        c1.C1();
        c2.C2();
    }
}
```

5a

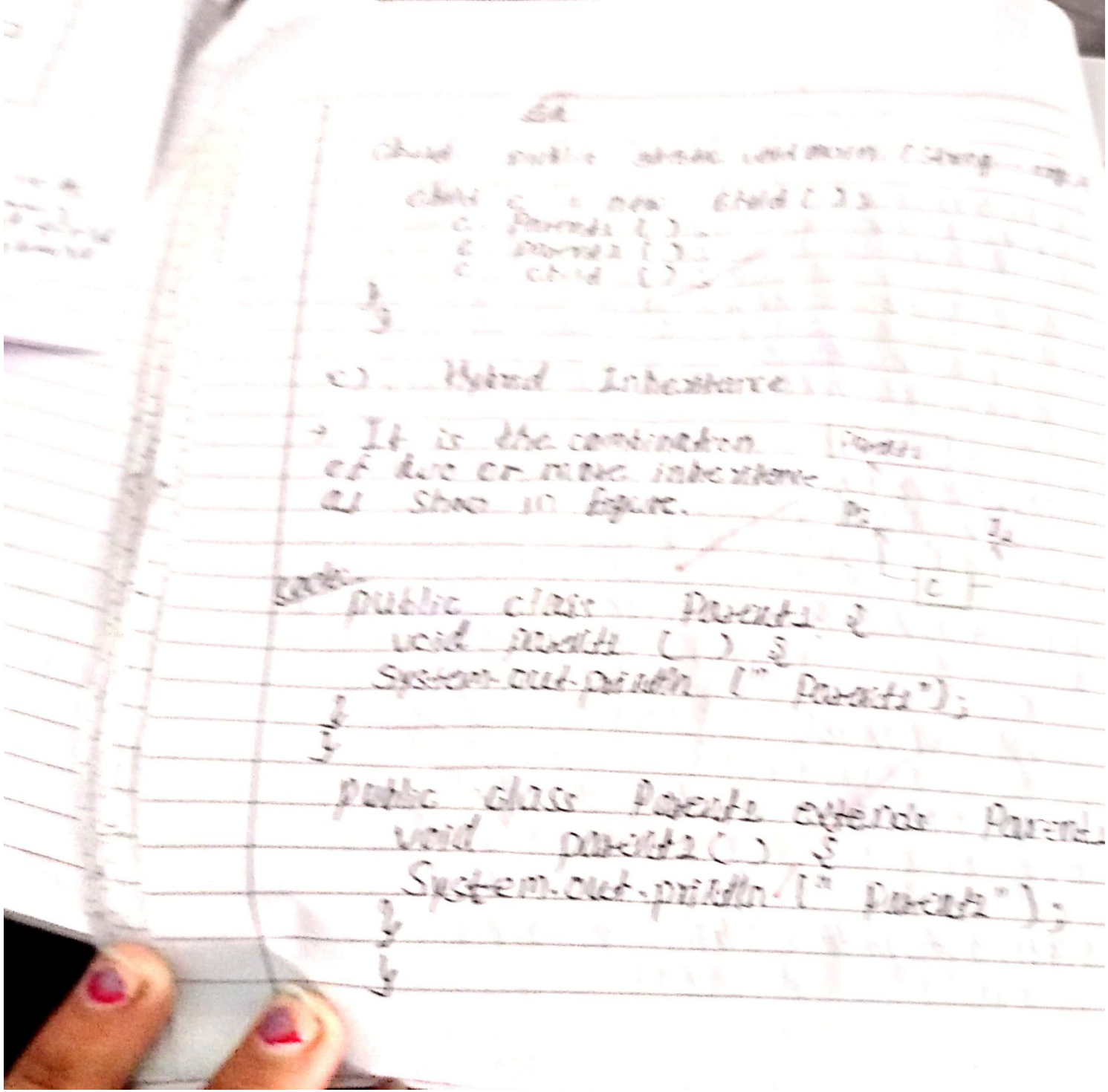
d) Multiple Inheritance

→ It has multiple parent of one is class and other are interfaces.



code:

```
public class Parent {
    void parent1 ( ) {
    System.out.println (" This is parent ");
    }

public interface ParentInterface {
    void parent2 ( );
    }

public Child extends Parent implements Child
{
    @ Override
    public void Parent2 ( )
    {
    System.out.println (" This is interface");
    }
    void child ( ) {
    System.out.println (" This is a child");
    }
```



c) Hybrid Inheritance

→ It is the combination of two or more inheritance as shown in figure.

code:

```
public class Parent1 {
    void fun1() {
    System.out.println("Parent1");
}
}

public class Parent2 extends Parent1 {
    void fun2() {
    System.out.println("Parent2");
}
}
```

5a

```
public interface I1 {
        void I1 ();
}
public child extends Parent2 implements I1
{
    @ Override
   public void I1 () {
    System.out.println (" Interface1");
   }
    void child ()
  {
   System.out.println (" child");
  }

  public static void main (String... args)
   {
     child c = new child ();
     c.parent1 ();
     c.parent2 ();
     c.I1 ();
     c.child ();
   }
  }
```

5b

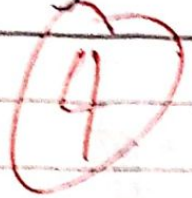
Use of collection framework:

→ It provides readymade architecture for manipulation and creating objects of ...
→ thread security
→ efficient data retrieval
→ similar types of algorithm
→ easy to implement

code example:

```
import java.util.Scanner ;
public class framework {
List<String> names = Arraylist < > ;
Scanner s = new Scanner (System.in);
void show ()
{
for (i= 0 ;i < 20; i++ )
{
System.in.println (" enter name of " + i +
                   " student " ) ;
String names = s.nextln ( );
names.add ( );
}
```

```
        System.out.println(" total no of student = "
                                    names.sizeof( ) );
    }
    public static void main (String... args)
    {
        Framework f = new Framework();
        f.show();
    }
}
```



Method Overloading vs Method Overriding:

→ Method Overloading has same method but different arguments within a class while method overriding has same method same return type and same arguments in a child class that has it's parent class.

example: Method Overloading.

```
public class Calculate {
    void add (int a, int b)
    {
        this.a = a;
        this.b = b;
    System.out.println (" sum = " + (a+b));
    }

    void add (int a, int b, float c)
    {
        this.a = a;
        this.b = b;
        this.c = c;
    System.out.println (" sum = " + (a+b+c));
```

National Academy of science and Technology
[Affiliated to Pokhara University]
Dhangadhi, Kailali (Nepal)
Accredited by University Grants Commission (UGC), Nepal (2022)

Additional answerbook No. : 3442
Number of additional answerbooks used : ..... Invigilator's Signature :
Name of the student : ..... Programme :
Roll No. (Class) : ..... Exam : ..... Grade/Semester : ..... Exam :
Paper : ..... Date :
* Students are required to write answer on both sides starting from the first page of the copy.
* This answerbook must be stiched properly with the original answerbook before delivering to the invigilator.

6a

```
}
public static void main (String... args){
    Calculate c = new Calculate();
    c.add(2,3);
    c.add(2,4,5.2f);
}
}
```

example: Method overriding

```
public class Arithmetic {
  void sub (int x, int y)
{
    this.a = a;
    this.b = b;
    System.out.println(" subtraction = " + (a-b));
}
}
public class AllArithmetic extends Arithmetic
{
```

(6a)

```
Class
void sub ( int x, int y )
{
  this.sub = sub : x
  this. y = y ;
  System.out.println (" subtraction = " +(x-y));
}
public static void main (Strings... args)
{
  AllArithmetic a : new AllArithmetic ( );
  a.sub ( );
}
}
```

(b)

Gb

final vs finally keywords

→ final keyword is used to make final or not changed to a class, member or method. It is always constant.

eg: final Pi = 3.14 ; (value can't be change);

eg: final class A {
} // class A cann't be inherit it's child class

eg: find void add ()
{ // function cann't be override
}

→ finally is a keyword used in exception handling. It executes at once although there is exception or not. It clean up the memory. It follow try-catch block.

finally {
try { } catch ( ) ;
}

4b.

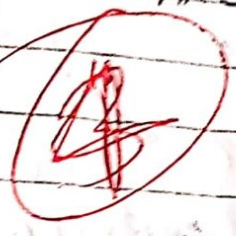
Looping statement in Software development is used for iterative process or repeat until the condition is fail.

example: for loop:

```
public class ForLoop {
    int i, sum = 0;
    void Sum() {
for (i = 1; i <= 20; i++)
{
    Sum = sum + i;
}
  System.out.println("sum =" +sum);
}
  public static void main (String... args)
{
    ForLoop f = new ForLoop();
        f.sum();
}
}
```

example: while loop

```
public class WhileLoop {
    int i = 5
    while (i!=0) void while() {
{
        System.out.println("i");
}
}
    public static void main (String... args)
{
        WhileLoop w = new WhileLoop();
        w.while();
}
}
```



6c. Upcasting and doon casting

Upcasting: It is the process to change child class to parent class type. It is done implicitly.

doowncasting: It is the process to change parent class to child class type.

Example:

```
public class Animal {

void sound (String name) {

System.out.println ("name" + sound);
}
}

public class Dog Extends Animal {
@ Override
void sound (String name) { meow"
System.out.println (" name " + "sound");
}
}

public class Cat extends Animal {
@ Overmde
void sound (String name) {
System.out.println (" name " + " bark ");
}
}
```

National Academy of science and Technology
[Affiliated to Pokhara University]
Dhangadhi, Kailali (Nepal)
Accredited by University Grants Commission (UGC), Nepal (2022)

Additional answerbook No. : 3952
Number of additional answerbooks used : .... Invigilator's Signature :
Name of the student : .... Programme :
Roll No. (Class) : .... Exam : .... Grade/Semester/ .... Exam :
Paper : .... Date :

Students are required to write answer on both sides starting from the first page of this copy
This answerbook must be stiched properly with the original answer book before delivering to the invigilator

6C

```
public UpdownDemo {
    public static void main (String.. args){

Animal d = new Dog ( );
Animal c = new Cat ( );
    d. sound ( );
    c. sound ( );

If ( d InstanceofDog )
{
   Dog d1 = (Dog)d ;
}
If ( c InstanceofCat )
{
  Cat c1 = (Cat) c ;
}
   d. sound ("Animal") ;
   c. sound ("Animal") ;
    }
  }
```

4

7 a

OOPs

It stands for Object Oriented Programming System which follows bottom to top approach. It is very popular programming paradigm. Eg Java, Python, PHP etc are OOPs.

Features

i) Class: Collection of similar type objects
ii) Object: Instance of class, real world entity

iii) Inheritance: Child class can get property of parent class.

iv) Abstraction: hiding of background from Security

v) Polymorphism: Once create many use. A function can be overload, override.

vi) Encapsulation: binding of data and behaviour at once

Q.6)

Features:

i) Class: Collection of similar type objects
ii) Object: Instance of class, real world entity

iii) Inheritance: Child class can get property of parent class.

iv) Abstraction: Hiding of background. Provides security.

v) Polymorphism: Once to wide many use. Different functions can be overload, override.

vi) Encapsulation: Binding of data and behaviour at once.

7C Access Modifier

Access Modifier restricts the scoped variable, method or class and others. It provides security. We can access them only that way that they are defind.

Access modifier	class	Package	Subclass	Global
private	✓	✗	✗	✗
default	✓	✓	✗	✗
protected	✓	✓	✓	✗
Public	✓	✓	✓	✓

Where, [✓] = accessible
[✗] = not accessible

Generally, in programming variables are private, method and constructors are public, In inheritance it follows protected.

We choose the access modifier as our requirements that keeps program more secure and effiecent.