1.a.
i. instance of a class

An object is a (software) bundle of related state and behavior.
State, Behavior, identity

ii. A class is the blueprint from which individual objects are created. –
Object is an instance of a class

iii. private, default, protected, public

b.
i. A class which is declared as abstract is known as an abstract class. It can
have abstract and non-abstract methods.
A class which hides the implementation details and shows only the functionality

```
abstract class Bike{
        abstract void start();
}
```

ii. An empty interface is known as tag or marker interface. These interfaces do not have any field and methods in it.

```
 Interface MyTaggable {   }
```

c.
i. to initialize the object
ii. Default constructor takes no parameters

```
Car {
 Car(){}

 Car (int x){}
}
```

initializes all instance variables to zero or null (def vals)

d. i.

```
Student stu = new Student();
Stu.setName("hjgfhfs");
```

ii. new Student().setName("shgjsgd");

iii. Perera
Same local variable and class variable name

name = name; → "this."

iv. stu.setName(“Priyantha”);

```
Public void setName(String name){
	this.name = name;
}
```

```
v. Student(String name, int age){
	this.name =name;
	this.age = age;
}
```

2. a. i. Polymorphism, method overloading

ii. Yes
no of arg

```
void sum(int a, int b){}
void sum(inta, int b, int c){}
```

DT args

```
Void Sum(int a, int b){}
Void Sum(float a, floatb){}
```

iii. class can give its own specific implementation to an inherited method without even modifying the parent class code

b. i. The get methods that allow a field to be viewed are known as accessor methods.

The set methods that allow a field to be changed are known as mutator
Methods

```
ii. public class Account{
	private String name;
	private double bal;

	public void setNmae(String name){
		this.name = name;
	}
	Public String getName(){
		Return name;
	}
}

class AccountDemo{
	psvm(String[] args){
		Account ac  = new Account();
		Ac.setName(“Kamal”);
```

```
                    String name = ac.getName();
                    Sout("sdfsdhf"+ac.getName());
          }
}
```

c. i. Inheritance

ii. code reusability, add/modify/change the behaviour (overriding),

```
Interface Shape{
          String color;
          Boolean filled;
}

Class Circle implements Circle{
          Double radius
}

Interface Rectangle extends Shape{
          Double width;
}
```

iv. upcasting/Implicit
Shape shp = new Circle();

Downcasting/ Explicit
Circle cir = (Circle) new Object();

3. a. I lot

ii. Compile without any error, runtime will get an exception

```
try{
          System.out.println(arr[7]);
}catch(ArraryINdexOut…./Exception e)
{
}
```

iii.

```
public void checkEligibility (double marks) throws NotEligible{

          If ( marks >= 80.00 )
                    Sout("Eligible");
           If ( marks < 80.00 )
                    throw new NotEligible("You are not Eligible");
```

}

B . i.
153

ii.
17234

4. a.i

ii.
Get compiled once and run it anywhere

double -→ Double

d i.
Compile error

Ii

```
public class ForLoopDemo {
            public static void main(String[] args){
                        int age = 10;
                        String names[] = { “Nimal”, “Kamal” };

                        for ( int i = 0, ~~age =10~~ ; i < names.length ; i++)
                        {
                                    String name = names[i];
                                    System.out.println(name + “ , “ + age);
                                    Age +=5;
                        }
            }
}
```

ii.

```
for (String name : names){
{
            System.out.println(name + “ , “ + age);
            Age +=5;
```

}

e. i.
Compile error

ii.
public **Person**(String name){
this.name = name;
//this.age =20;
this(20)
}

Public Person(int age){
 this.age =age;
}