1. Predict the output:

```
abstract class A{
private  int a;
abstract void show();
//getters setters constructor available
}
class B extends A{
    private int b;
    //getters setters constructor available
    void show(){
        System.out.println("B's show");
    }
}
class C extends A{
    private int c;
    //getters setters constructor available
    void show(){
        System.out.println("C's show");
    }
}
public class CastingTest {
    public static void main(String[] args) {
    A aref=new B();
    C c=(C)aref;
    c.show();
    }
}
```

2. What's the problem in below code?

```
    class Account{
    private  int accId;
    Account(int accId){
        this.accId=accId;
    }
    }
    class SavingAccount extends Account{
        private int roi;
        SavingAccount(int roi){
            this.roi=roi;
        }
    }
    class CurrentAccount extends Account{
        private int overdraftLimit;
        CurrentAccount(int overdraftLimit)
        {
            this.overdraftLimit=overdraftLimit;
        }
    }
    public class CastingTest {
        public static void main(String[] args) {
            Account account=new SavingAccount(2);}
    }
```

3. Consider following class:

```
       class Message{
           private int msgId;
           private String message;
           Message(int msgId,String  message)
           {
               this.msgId=msgId;
               this.message=message;
           }
```

```
        }
```

Consider following code snippet:

```
public class TestConcept {
    public static void main(String[] args) {
        Message m1=new Message(101,"Greetings");
        Message m2=new Message(101,"GREETINGS");
        System.out.println(m1.equals(m2));
    }
}
```

The above code results **false. Why?**

What changes you'll made in the Message class to produce the output as **True.**

4. What will be the output of following code:

```
final class Calculation{
    public final void doCalculation(){
        //some stuff goes here
              System.out.println("hello");
    }
    public void doStuff(){
        //some stuff goes here
       System.out.println("hi dostuff");
    }
}
class MoreCalculation extends Calculation{
    public void doCalculation(){
        //some stuff goes here
    }
    public static void main(String[] args) {
        Calculation calculation=new MoreCalculation();
        calculation.doCalculation();
    }
}
```

5. Can we have private constructor in class? What is the purpose of such constructor?
6. Consider the below code:

```
       abstract class ProjectTemplate
       {  //some required attributes
           public void buildProject(){
               getRequirement(); //step 1
               doPlanning();  //step 2
               doDesigning();  //step 3
               doBuilding();  //step 4
               doTesting();  //step 5
           }
           public abstract void getRequirement();
           public abstract void doPlanning();
           public abstract void doDesigning();
           public abstract void doBuilding();
           public abstract void doTesting();
       }
```

How can we make sure that the subclass will follow the same order while building any project.

7. Assuming following classes:

```
    abstract class Account{
        private int accId;
        Account(int accId)
        {this.accId=accId;}
    }
```

```
class Saving extends Account{
    private float roi;
    Saving(int accId,float roi)
    {super(accId);
    this.roi=roi;}
    }

class Current extends Account{
    private int  overdraft;
    Current(int accId,int overdraft)
    {
        super(accId);
        this.overdraft=overdraft;
    }
}
public class Sample {
    static void test(Account[] accounts)
    {
      //add statements to count number of each account in passed array
    }
    public static void main(String[] args) {
     Account accounts[]={new Saving(11011,2.3f),new Current(11023,1500)};
     test(accounts);
    }
}
```

8. When do we need to downcast object? Explain with code segment.
9. Consider the below code:

```
    class Hello{
        private String msg;
       Hello(String msg)
       {
           this.msg=msg;
       }
        @Override
        public String toString() {
           super.toString();
            return "Hello{" +
                    "msg='" + msg + '\'' +
                    '}';
        }
    }
```

What will be the output of below code snippet:

```
public class Test {
    public static void main(String[] args) {
        Hello hello=new Hello("All Well!!");
        System.out.println(hello);
    }
}
```

10. Write class structure to manage batches of multiple people. Each batch does multiple tasks.
    People has their own attributes like id,name,contact details etc.
    Each task has taskId,name,taskLeadBy,taskCompletionDate,taskStartDate
    Task can be of two types either Session task or Assessment task. Session task has attributes like sessionHours,sessionTopic etc. Assessment task has attributes like assessmentMarks,assessmentRemarks etc..