Questions

1. What exception types can be caught by the following handler?

```
try {
    //some codes
}
catch (Exception e) {
   //some codes
}
```

   Is this exception handler recommended?

2. Is there anything wrong with the following exception handler as written? Will this code compile?

```
try {
      //some codes that will throw ArithmeticException
}
catch (Exception e) {
      //some codes
}
catch (ArithmeticException a) {
      //some codes
}
```

3. What will be the output of the following programs?

```
public class X
{
    public static void main(String [] args)
    {
        try {
            badMethod();
            System.out.print("A");
        }
        catch (RuntimeException ex) {
            System.out.print("B");
        }
        catch (Exception ex1) {
            System.out.print("C");
        }
        System.out.print("E");
    }
    public static void badMethod(){
        throw new RuntimeException();
    }
}
```

```
public class RTExcept
{
    public static void throwit ()
    {
        System.out.print("C ");
        throw new RuntimeException();
    }
    public static void main(String [] args)
    {
        try
        {
            System.out.print("A ");
            throwit();
        }
        catch (Exception re )
        {
            System.out.print("B ");
        }
        System.out.println("D ");
    }
}
```

```
public class Test
{
    public static void aMethod() throws Exception
    {
        try
        {
            throw new Exception();
        }
        catch(NullPointerException ex) {
             System.out.print("in NullPointerException");
        }
        finally
        {
            System.out.print("finally ");
        }
    }
    public static void main(String args[])
    {
        try
        {
            aMethod();
        }
        catch (Exception e)
        {
            System.out.print("exception ");
        }
        System.out.print("finished");
    }
}
```

```
public class X
{
    public static void main(String [] args)
    {
        try
        {
            badMethod();
            System.out.print("A");
        }
        catch (Exception ex)
        {
            System.out.print("B");
        }
        finally
        {
            System.out.print("C");
        }
        System.out.print("D");
    }
    public static void badMethod() {}
}
```

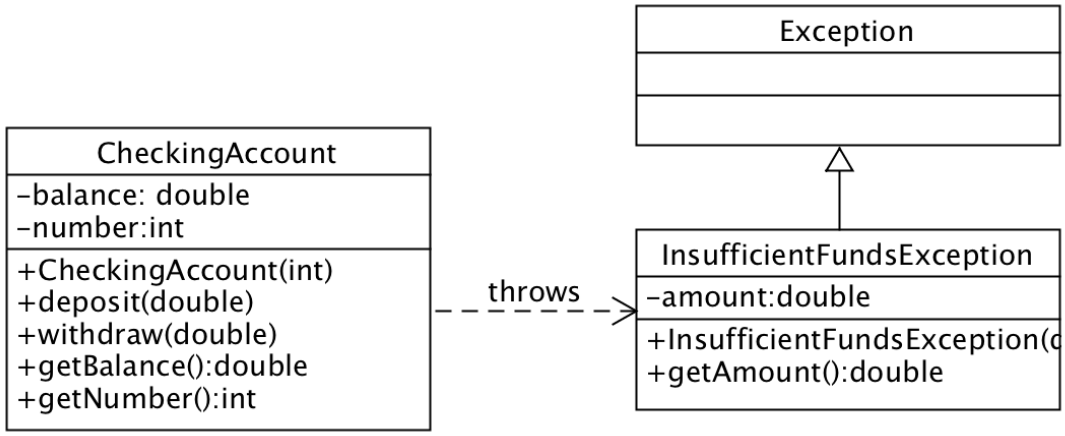
```
public class X
{
    public static void main(String [] args)
    {
        try
        {
            badMethod();
            System.out.print("A");
        }
        catch (Exception ex)
        {
            System.out.print("B");
        }
        finally
        {
            System.out.print("C");
        }
        System.out.print("D");
    }
    public static void badMethod()
    {
        throw new RuntimeException();
    }
}
```

```
class Exc0 extends Exception { }
class Exc1 extends Exc0 { }

public class Test{
    public static void main(String args[]) {
        try {
            throw new Exc1();
        }
        catch (Exc0 e0) {
            System.out.println("Ex0 caught");
        }
        catch (Exception e) {
            System.out.println("exception caught");
        }
    }
}
```

4. Create your own Exception

   Given the following class BankDemo and the class design.
   CheckingAccount class withdraw() method will throw InsufficientFundsException if there is not enough fund in the account.
   Write the CheckingAccount and InsufficientFundsException class




```
public class BankDemo
{
   public static void main(String [] args)
   {
      CheckingAccount c = new CheckingAccount(101);
      System.out.println("Depositing $500...");
      c.deposit(500.00);
      try
      {
         System.out.println("\nWithdrawing $100...");
         c.withdraw(100.00);
         System.out.println("\nWithdrawing $600...");
         c.withdraw(600.00);
      }catch(InsufficientFundsException e)
      {
         System.out.println("Sorry, but you are short $"
```

```
                + e.getAmount());

        }
    }
}
```