Revision

- Explain the meaning of each word/special character.
- Also explain how the following statements are executed.

```
Student amy = new Student();
amy.setName("Amy Chan");
```

```
public class Student {
  private String name;
  ...
  public void setName(String aName) {
    name = aName;
  }
}
```


Lab 02 Issues

Customer-2.java

```
1  public class Customer{
2
3  String name;
4  int age;
5  String inputName;
6  String inputAge;
7  //setter
8  public void setName(String inputName){
9  name= inputName;
10 }
11 public void setAge(int inputAge){
12 age = inputAge;
13 }
14
15 //getter
16 public String getName(){
17 return name;
18 }
19
20 public int getAge(){
21 return age;
22 }
23
24 }
```

- only .java files are needed (no .class file, no .doc file)
- indentation
- "private" for attribute
- no need to declare parameter variables

Lab 02 : Good version

Customer-4.java

```
1  public class Customer {
2      private String name;
3      private int age;
4
5      //setter
6      public void setName(String inputName) {
7          name = inputName;
8      }
9      public void setAge(int inputAge) {
10         age = inputAge;
11     }
12
13     //getter
14     public String getName() {
15         return name;
16     }
17     public int getAge() {
18         return age;
19     }
20 }
```


Lab 02 : Issues on testing class

TestCustomer-5.java

```
1  public class TestCustomer {
2      public static void main(String[] args) {
3         Customer joseph;
4         Customer billy;
5
6        joseph = new Customer ();
7        billy = new Customer ();
8
9        joseph.setName("Joseph Chan");
10       joseph.setAge(20);
11       billy.setName("Billy Wong");
12       billy.setAge(20);
13
14     System.out.println("The customer no.001 name:" + joseph.getName());
15     System.out.println("He is " + joseph.getAge());
16     System.out.println("The customer no.002 name:" + billy.getName());
17     System.out.println("He is " + billy.getAge());
18
19    }
20 }
```


Error messages

- `File : lineNumber : error message`

position

HelloWorld.java

```
1  public class HelloWorld {
2    public static void main(String[] args) {
3      System.out.println("Hello World")
4    }
5  }
```

```
HelloWorld.java:3: ';' expected
    System.out.println("Hello World")
                                      ^
1 error

> Terminated with exit code 1.
```


Transfer money using objects

- Want to: transfer $100 from account A to account B
- model account: balance (simplest)
- model transfer (debit account A and credit account B)

Model Account

Account class

balance : attribute
getBalance( ) : method
setBalance( ) : method
transferTo( ) : method

```
public class Account {
  private double balance;
  public double getBalance() { ... };
  public void setBalance(double aBalance) { ... };
  public void transferTo(Account ac, double amount)
     {...}
}
```

accountA: object

balance = 500
getBalance( )
setBalance( )
transferTo( )

accountB: object

balance = 200
getBalance( )
setBalance( )
transferTo( )

```
Account accountA = new Account();
accountA.setBalance(500);
```

```
Account accountB = new Account();
accountB.setBalance(200);
```


Model Transfer

After transfering
$100 from accountA
to accountB

accountA : object

balance = 400
getBalance( )
setBalance( )
transferTo( )

accountB: object

balance = 300
getBalance( )
setBalance( )
transferTo( )

```
accountA.transferTo(accountB, 100);
```

```
public class Account {
  private double balance;
  public void transferTo(Account ac, double amount) {
    balance -= amount;
    double newBalance = ac.getBalance() + amount;
    ac.setBalance(newBalance);
  }
...
```


Application: Bank

```
public class Bank {
  public static void main(String[] args) {
     Account accountA = new Account();
     accountA.setBalance(500);

     Account accountB = new Account();
     accountB.setBalance(200);

     accountA.transferTo(accountB, 100);

  }
}
```


Do all in main(): similar to C

```
public class Bank {
  public static void main(String[] args) {
    double accountABalance = 500;
    double accountBBalance = 200;
    accountABalance -= 100;
    accountBBalance += 100;
  }
}
```

```
public class Bank {
  public static void main(String[] args) {
     Account accountA = new Account();
     accountA.setBalance(500);

     Account accountB = new Account();
     accountB.setBalance(200);

     accountA.transferTo(accountB, 100);

  }
}
```

- model balance only: shorter
- transfer: longer

If balance stored using String

```
public class Bank {
  public static void main(String[] args) {
    String accountABalance = "500";
    String accountBBalance = "200";
    accountABalance = "" + (Double.parseDouble(accountABalance ) – 100);
    accountBBalance = "" + (Double.parseDouble(accountBBalance ) + 100);
  }
}
```

- all statements need to be changed
- usually application much longer (e.g. 3000 lines)

If balance stored using String

```
public class Bank {
  public static void main(String[] args) {
     Account accountA = new Account();
     accountA.setBalance(500);

     Account accountB = new Account();
     accountB.setBalance(200);

     accountA.transferTo(accountB, 100);


  }
}
```

- no statement change except in Account class (centralized)

```
public class Account {
  private String balance;
  public double getBalance() {
    return Double.parseDouble(balance);
  };
  public void setBalance(double aBalance) {
    balance = "" + aBalance;
  };
  ...
}
```