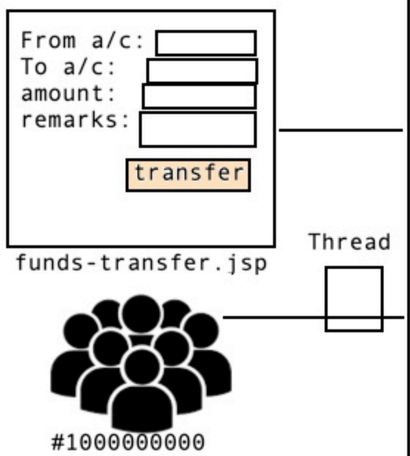
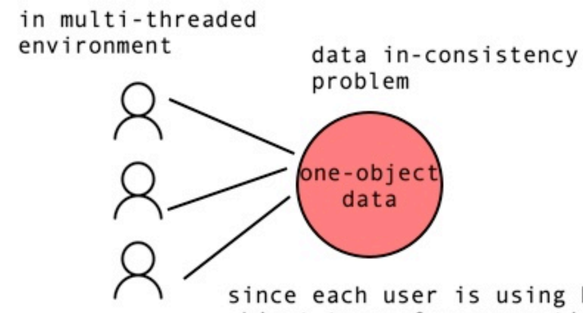
#1



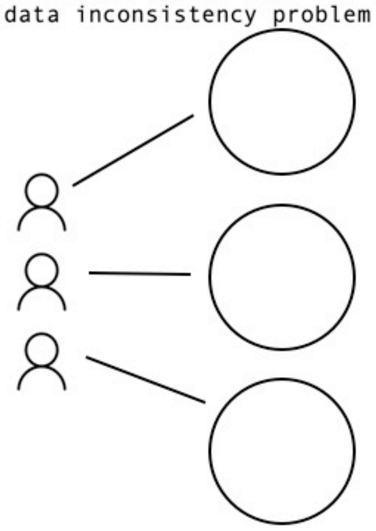

```
@WebServlet("/fundsTransfer")
class FundsTransferServlet extends HttpServlet {
  String fromAccount;
  String toAccount;
  double amount;
  String remarks;
  double fromAccountBalance;
  double toAccountBalance;

  public void service(httpReq, httpResp) throws ServletException {
    String fromAccount;
    String toAccount;
    double amount;
    String remarks;
    double fromAccountBalance;
    double toAccountBalance;

    fromAccount = httpReq.getParameter("fromAccount");
    toAccount = httpReq.getParameter("toAccount");
    amount = Double.parseDouble(httpReq.getParameter("amount");
    remarks = httpReq.getParameter("remarks");


    // fetch the balance of fromAccount from database
    fromAccountBalance = … fetch from db
    toAccountBalance = … fetch from db
    if(fromAccountBalance < amount) {
      throw new InsufficientFundsException("insufficient balance for transfer");
    }
    fromAccountBalance = fromAccountBalance - amount;
    toAccountBalance = toAccountBalance + amount;
    // update the balances of both accounts in the database
    fromAccount with fromAccountBalance … store in db
    toAccount with toAccountBalance … store in db
    httpReq.getRequestDispatcher("/funds-transfer-success.jsp").
forward(httpReq, httpResp);
  }
}
```

1. How many objects of the Servlet will be created by the Servlet container?
always the servlet container will creates only one object per each servlet we configured in web.xml and reuses the same object for serving any number of user requests that are coming for that servlet.
creating request-per-object increases the consumption of jvm memory and quickly the jvm will runs out of memory that leads to server crash, to avoid this the ServletContainer reuses the same object for all the requests it has received for it.
A Servlet executes in an Multi-Threaded model
1 req = 1 thread of execution -> [single Servlet object]
1 req = 2 thread of execution -> same object of Servlet

2. is an Servlet is Singleton class?
The Servlet by itself is not Singleton, we can create any number of objects for a Servlet class using new operator. but technical stand-point of view, since the ServletContainer creates only one object, it logically acts as Singleton



since each user is using his own object to perform operation, even in multi-threaded env we will not run into data inconsistency problem



```
class WebUser implements Runnable {
  Date dob;
  public WebUser(Date dob) {
    this.dob = dob;
  }
  public void run() {
    AgeCalculator ageCalculator = AgeCalculator.getInstance();
    //ageCalculator.setDob(dob);
    int age = ageCalculator.age(dob);
    sop(age);
  }
}
```

```
T1
stack
leln = 10
dob=1990

T2
stack
cp=10
dob=1980
```

```
class AgeCalculator {
  private static AgeCalculator instance;

  private AgeCalculator() {}

  public synchronized static AgeCalculator getInstance() {
    if(instance == null) {
       instance = new AgeCalculator();
    }
    return instance;
  }

  public int age(Date dob) {
    Date now = new Date();
    int days=now - dob;
    return days;
  }
}
```

Test.java

```
WebUser user1 = new WebUser(new Date(1990,0,1);
WebUser user2 = new WebUser(new Date(1980,0,1);
WebUser user3 = new WebUser(new Date(1970,0,1);

new Thread(user1).start();
new Thread(user2).start();
new Thread(user3).start();
```

```
How to make an object thread-safe?
singleton:
if it is an singleton-class and multiple threads of execution are sharing the same object then:
1. the object should not hold any non-sharable state, all the state the object holds should be sharable only and the access to the sharable state should be synchronized
2. if the object holds the state, it must be either read-only
3. if the object holds an non-sharable state, dont declare the state as attributes, rather declare them as parameters/local variables within the method in which we use the data
to perform operation.

non-singleton:
if it is an non-singleton object: then use one object per one-thread of execution
```