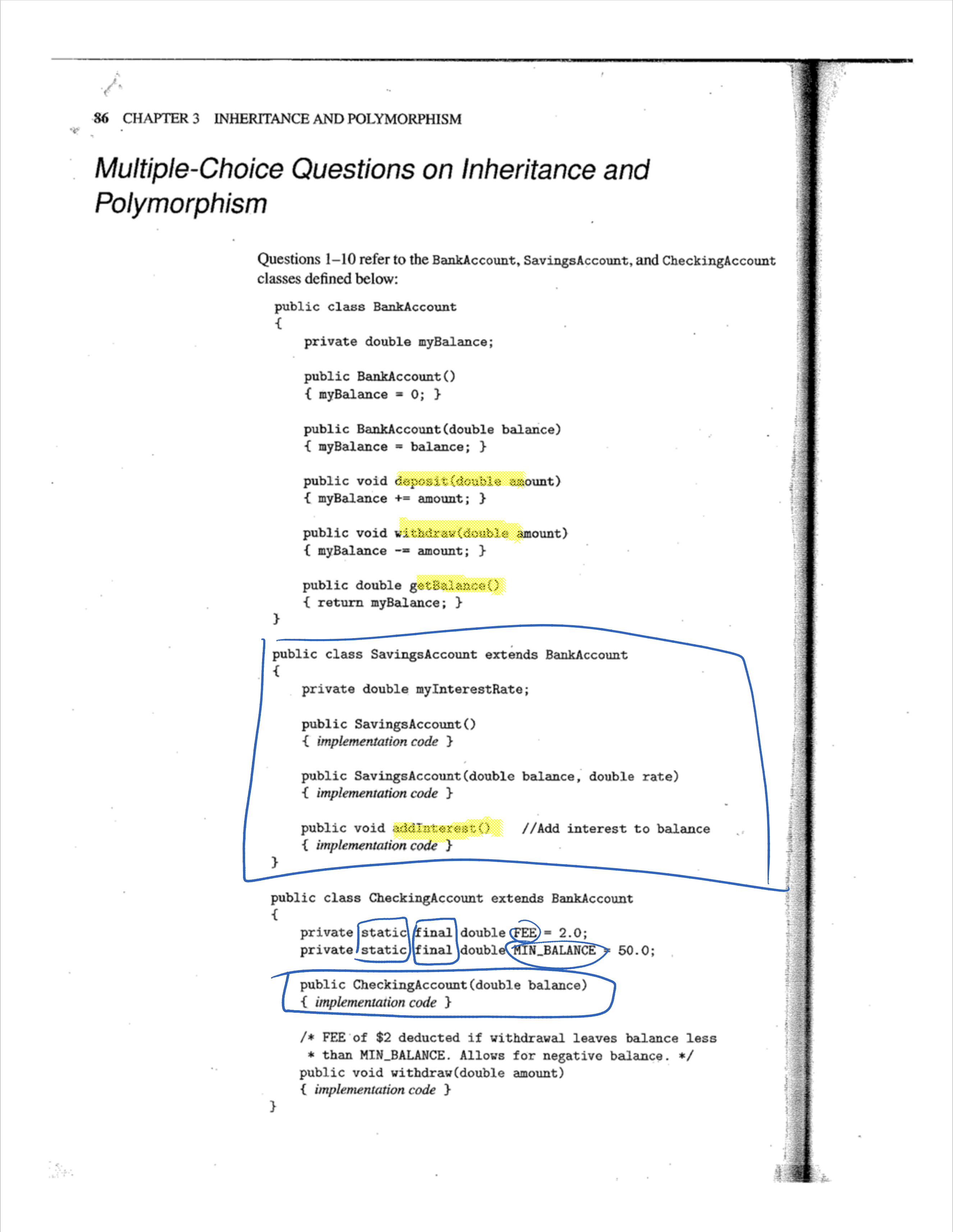
Ch 7 Multi-Choice Review

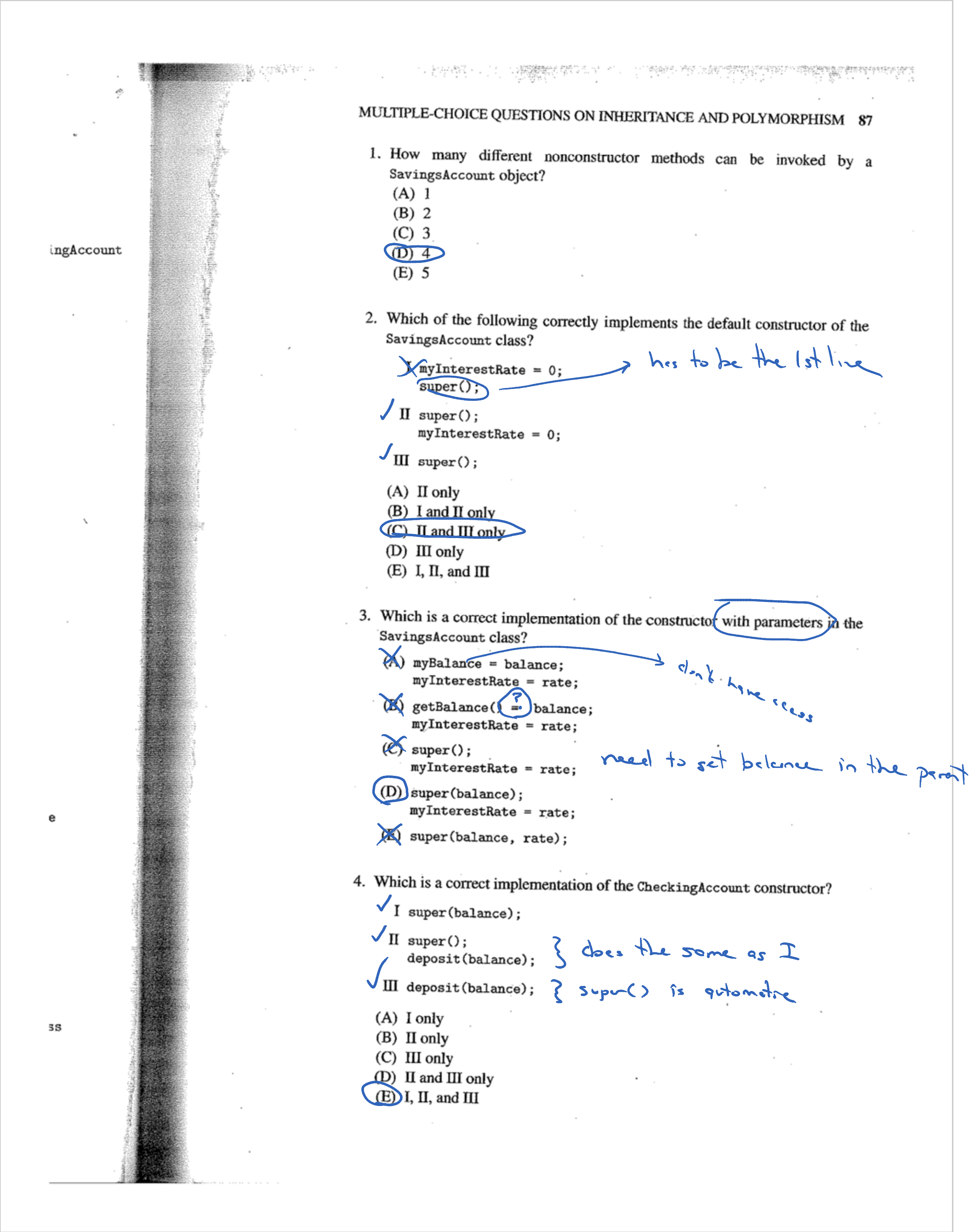
Wednesday, March 06, 2013
8:02 AM



Multiple-Choice Questions on Inheritance and Polymorphism

Questions 1–10 refer to the BankAccount, SavingsAccount, and CheckingAccount classes defined below:

```
public class BankAccount
{
    private double myBalance;

    public BankAccount()
    { myBalance = 0; }

    public BankAccount(double balance)
    { myBalance = balance; }

    public void deposit(double amount)
    { myBalance += amount; }

    public void withdraw(double amount)
    { myBalance -= amount; }

    public double getBalance()
    { return myBalance; }
}

public class SavingsAccount extends BankAccount
{
    private double myInterestRate;

    public SavingsAccount()
    { implementation code }

    public SavingsAccount(double balance, double rate)
    { implementation code }

    public void addInterest()     //Add interest to balance
    { implementation code }
}

public class CheckingAccount extends BankAccount
{
    private static final double FEE = 2.0;
    private static final double MIN_BALANCE = 50.0;

    public CheckingAccount(double balance)
    { implementation code }

    /* FEE of $2 deducted if withdrawal leaves balance less
     * than MIN_BALANCE. Allows for negative balance. */
    public void withdraw(double amount)
    { implementation code }
}
```



ingAccount

e

ss

1. How many different nonconstructor methods can be invoked by a `SavingsAccount` object?
   (A) 1
   (B) 2
   (C) 3
   [✓ (D) 4]
   (E) 5

2. Which of the following correctly implements the default constructor of the `SavingsAccount` class?

```
   I   myInterestRate = 0;
       super();
```
has to be the 1st line

```
✓ II  super();
      myInterestRate = 0;

✓ III super();
```

   (A) II only
   (B) I and II only
   [✓ (C) II and III only]
   (D) III only
   (E) I, II, and III

3. Which is a correct implementation of the constructor with parameters in the `SavingsAccount` class?

```
   (A) myBalance = balance;
       myInterestRate = rate;
```
don't have access

```
   (B) getBalance() = balance;
       myInterestRate = rate;
```
?

```
   (C) super();
       myInterestRate = rate;
```
need to set balance in the parent

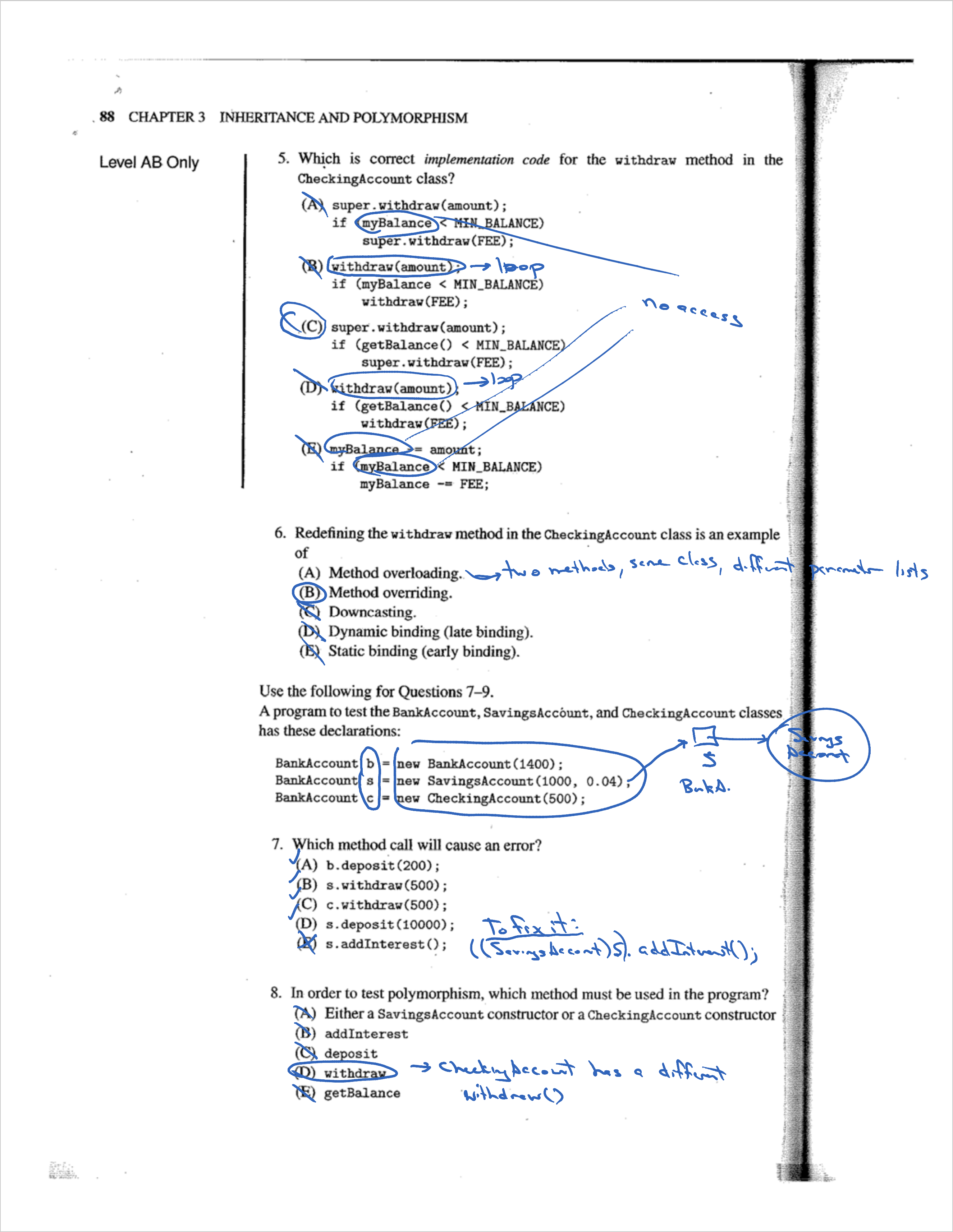
```
   [✓ (D)] super(balance);
       myInterestRate = rate;

   (E) super(balance, rate);
```

4. Which is a correct implementation of the `CheckingAccount` constructor?

```
✓ I   super(balance);

✓ II  super();
      deposit(balance);
```
} does the same as I

```
✓ III deposit(balance);
```
} super() is automatic

   (A) I only
   (B) II only
   (C) III only
   (D) II and III only
   [✓ (E)] I, II, and III



Level AB Only

5. Which is correct *implementation code* for the `withdraw` method in the `CheckingAccount` class?

(A)
```
super.withdraw(amount);
if (myBalance < MIN_BALANCE)
    super.withdraw(FEE);
```
(B)
```
withdraw(amount);   → loop
if (myBalance < MIN_BALANCE)
    withdraw(FEE);
```
no access

[✓ (C)]
```
super.withdraw(amount);
if (getBalance() < MIN_BALANCE)
    super.withdraw(FEE);
```
(D)
```
withdraw(amount);   → loop
if (getBalance() < MIN_BALANCE)
    withdraw(FEE);
```
(E)
```
myBalance -= amount;
if (myBalance < MIN_BALANCE)
    myBalance -= FEE;
```

6. Redefining the `withdraw` method in the `CheckingAccount` class is an example of

(A) Method overloading. → two methods, same class, different parameter lists

[✓ (B)] Method overriding.

(C) Downcasting.

(D) Dynamic binding (late binding).

(E) Static binding (early binding).

Use the following for Questions 7–9.

A program to test the `BankAccount`, `SavingsAccount`, and `CheckingAccount` classes has these declarations:

```
BankAccount b = new BankAccount(1400);
BankAccount s = new SavingsAccount(1000, 0.04);
BankAccount c = new CheckingAccount(500);
```

s → Savings Account

BankA.

7. Which method call will cause an error?

✓ (A) `b.deposit(200);`

✓ (B) `s.withdraw(500);`

✓ (C) `c.withdraw(500);`

(D) `s.deposit(10000);`

(E) `s.addInterest();`

To fix it:

((SavingsAccount)s).addInterest();

8. In order to test polymorphism, which method must be used in the program?

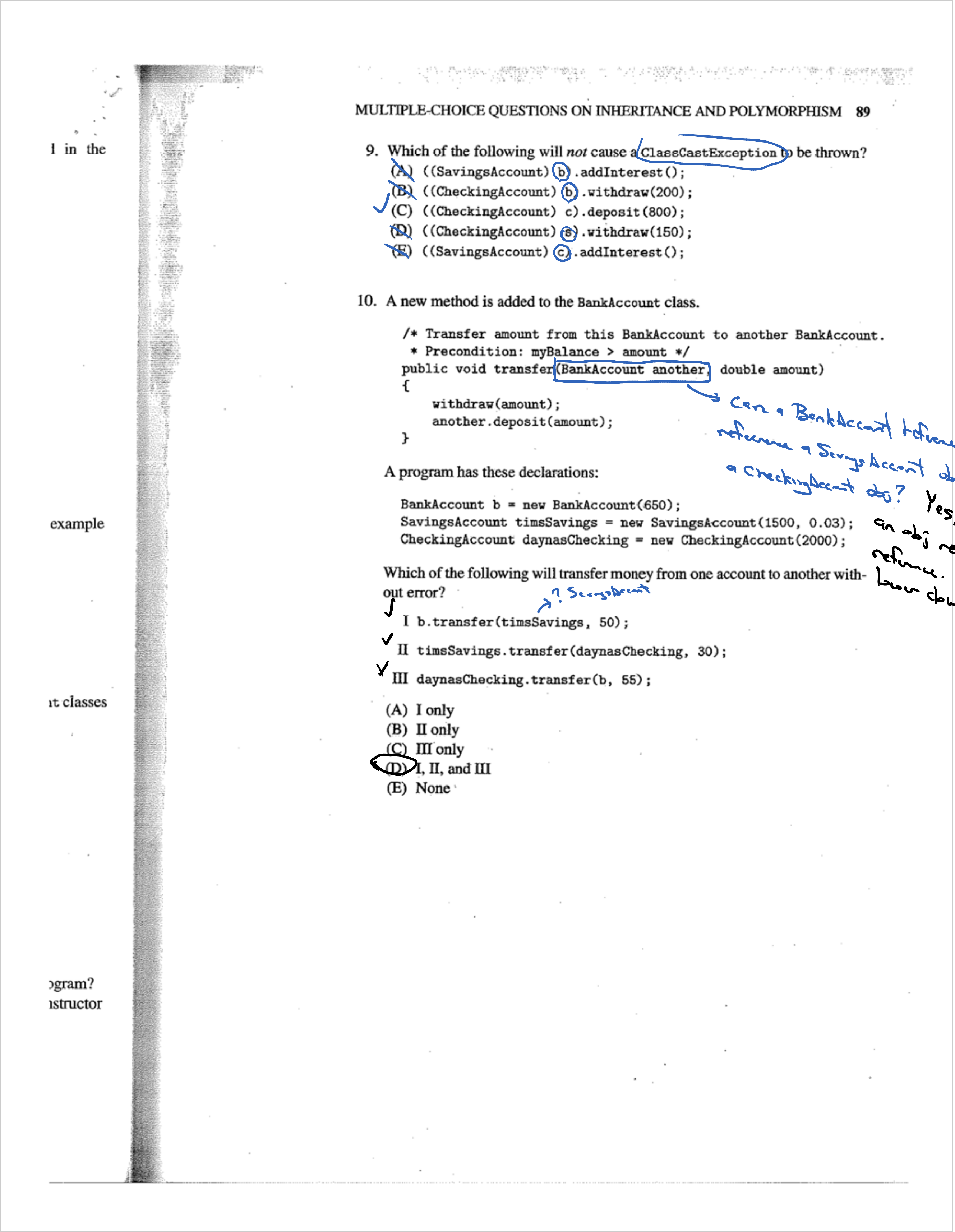
(A) Either a `SavingsAccount` constructor or a `CheckingAccount` constructor

(B) `addInterest`

(C) `deposit`

[✓ (D)] `withdraw` → CheckingAccount has a different withdraw()

(E) `getBalance`



l in the

example

ıt classes

ɔgram?
ıstructor

9. Which of the following will *not* cause a `ClassCastException` to be thrown?

   (A) `((SavingsAccount) b).addInterest();`
   (B) `((CheckingAccount) b).withdraw(200);`
   ✓ (C) `((CheckingAccount) c).deposit(800);`
   (D) `((CheckingAccount) s).withdraw(150);`
   (E) `((SavingsAccount) c).addInterest();`

10. A new method is added to the `BankAccount` class.

```
/* Transfer amount from this BankAccount to another BankAccount.
 * Precondition: myBalance > amount */
public void transfer(BankAccount another, double amount)
{
    withdraw(amount);
    another.deposit(amount);
}
```

→ Can a BankAccount ~~refere~~ reference a SavingsAccount obj? a CheckingAccount obj?

Yes, because an obj ref can reference something lower down the tree.

A program has these declarations:

```
BankAccount b = new BankAccount(650);
SavingsAccount timsSavings = new SavingsAccount(1500, 0.03);
CheckingAccount daynasChecking = new CheckingAccount(2000);
```

Which of the following will transfer money from one account to another without error?

? SavingsAccount

✓ I `b.transfer(timsSavings, 50);`

✓ II `timsSavings.transfer(daynasChecking, 30);`

✓ III `daynasChecking.transfer(b, 55);`

(A) I only
(B) II only
(C) III only
[✓ (D) I, II, and III]
(E) None

11. Consider these class declarations:

```
public class Person
{
    ...
}

public class Teacher extends Person
{
    ...
}
```

Constructors don't inherit

Which is a true statement?

~~I~~ Teacher inherits the constructors of Person.
[✓ II] Teacher can add new methods and private instance variables.
[✓ III] Teacher can override existing private methods of Person.

worded poorly because private methods do not inherit, so it's technically not overriding

(A) I only
(B) II only
(C) III only
(D) I and II only
[✓ (E)] II and III only

12. Which statement about abstract classes and interfaces is *false*?

✓ (A) An interface cannot implement any methods, whereas an abstract class can.
✓ (B) A class can implement many interfaces but can have only one superclass.
✓ (C) An unlimited number of unrelated classes can implement the same interface.
✓ (D) It is not possible to construct either an abstract class object or an interface object.
[✓ (E)] All of the methods in both an abstract class and an interface are public.

can have private methods