Questions 1–10 refer to the BankAccount, SavingsAccount, and CheckingAccount

defined below:

```
public class BankAccount
{
    private double balance;

    public BankAccount()
    { balance = 0; }

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

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

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

    public double getBalance()
    { return balance; }
}

public class SavingsAccount extends BankAccount
{
    private double interestRate;

    public SavingsAccount()
    { /* implementation not shown */ }

    public SavingsAccount(double acctBalance, double rate)
    { /* implementation not shown */ }

    public void addInterest()     //Add interest to balance
    { /* implementation not shown */ }
}

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

    public CheckingAccount(double acctBalance)
    { /* implementation not shown */ }

    /** FEE of $2 deducted if withdrawal leaves balance less
     *  than MIN_BALANCE. Allows for negative balance. */
    public void withdraw(double amount)
    { /* implementation not shown */ }
}
```

1. Of the methods shown, 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   interestRate = 0;
    super();

II  super();
    interestRate = 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) balance = acctBalance;
    interestRate = rate;

(B) getBalance() =  acctBalance;
    interestRate = rate;

(C) super();
    interestRate = rate;

(D) super(acctBalance);
    interestRate = rate;

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

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

```
I   super(acctBalance);

II  super();
    deposit(acctBalance);

III deposit(acctBalance);
```

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

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

   (A)
   ```
   super.withdraw(amount);
   if (balance < MIN_BALANCE)
       super.withdraw(FEE);
   ```
   (B)
   ```
   withdraw(amount);
   if (balance < MIN_BALANCE)
       withdraw(FEE);
   ```
   (C)
   ```
   super.withdraw(amount);
   if (getBalance() < MIN_BALANCE)
       super.withdraw(FEE);
   ```
   (D)
   ```
   withdraw(amount);
   if (getBalance() < MIN_BALANCE)
       withdraw(FEE);
   ```
   (E)
   ```
   balance -= amount;
   if (balance < MIN_BALANCE)
       balance -= FEE;
   ```

6. Redefining the `withdraw` method in the `CheckingAccount` class is an example of
   (A) method overloading.
   (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:

[illegible]
```
BankAccount b = new BankAccount(1400);
BankAccount s = new SavingsAccount(1000
```