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 }
}
```

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

   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;
   ```

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

   (C)
   ```
   super();
   myInterestRate = rate;
   ```

   (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);
   ```

   III
   ```
   deposit(balance);
   ```

   (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 (myBalance < MIN_BALANCE)
    super.withdraw(FEE);
```

(B)
```
withdraw(amount);
if (myBalance < 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)
```
myBalance -= amount;
if (myBalance < MIN_BALANCE)
    myBalance -= 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:

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

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();`

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`
   (E) `getBalance`

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);
}
```

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?

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