MULTIPLE-CHOICE QUESTIONS ON INHERITANCE AND POLYMORPHISM

Questions 1–9 refer to the BankAccount, SavingsAccount, and CheckingAccount classes 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 and 8.

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

```
/** Transfer amount from this BankAccount to another BankAccount.
 *  Precondition: balance > amount
 *  @param another a different BankAccount object
 *  @param amount the amount to be transferred
 */
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

10. Consider these class declarations.

```
public class Person
{
    ...
}

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

Which is a true statement?

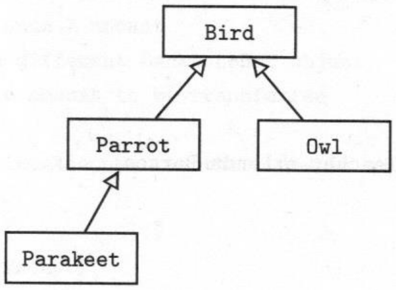
I `Teacher` inherits the constructors of `Person`.
II `Teacher` can add new methods and private instance variables.
III `Teacher` can override existing private methods of `Person`.

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

11. Which statement about subclass methods is false?

(A) Writing two subclass methods with the same name but different parameters is called method overriding.
(B) A public method in a subclass that is not in its superclass is not accessible by the superclass.
(C) A private method in a superclass is not inherited by its subclass.
(D) Two different subclasses of the same superclass inherit the same methods of the superclass.
(E) If `Class1` is a superclass of `Class2`, and `Class2` is a superclass of `Class3`, and `Class2` has no overridden methods, `Class3` inherits all the public methods of `Class1`.

12. Consider the following hierarchy of classes.



A program is written to print data about various birds:

```
public class BirdStuff
{
    public static void printName(Bird b)
    { /* implementation not shown */ }

    public static void printBirdCall(Parrot p)
    { /* implementation not shown */ }

    //several more Bird methods

    public static void main(String[] args)
    {
        Bird bird1 = new Bird();
        Bird bird2 = new Parrot();
        Parrot parrot1 = new Parrot();
        Parrot parrot2 = new Parakeet();
        /* more code */
    }
}
```

Assuming that all of the given classes have default constructors, which of the following segments of `/* more code */` will cause an error?

(A) `printBirdCall(bird2);`
(B) `printName(parrot2);`
(C) `printName(bird2);`
(D) `printBirdCall(parrot2);`
(E) `printBirdCall(parrot1);`

Refer to the classes below for Questions 13 and 14.

```
public class ClassA
{
    //default constructor not shown ...

    public void method1()
    { /* implementation of method1 */ }

    public void method2()
    { /* implementation of method2 */ }
}

public class ClassB extends ClassA
{
    //default constructor not shown ...

    public void method1()
    { /* different implementation from method1 in ClassA*/ }

    public void method3()
    { /* implementation of method3 */ }
}
```

13. The `method1` method in `ClassB` is an example of
    (A) method overloading.
    (B) method overriding.
    (C) polymorphism.
    (D) data encapsulation.
    (E) procedural abstraction.

14. Consider the following declarations in a client class.

```
ClassA ob1 = new ClassA();
ClassA ob2 = new ClassB();
ClassB ob3 = new ClassB();
```

Which of the following method calls will cause an error?

I `ob1.method3();`

II `ob2.method3();`

III `ob3.method2();`

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

Use the declarations below for Questions 15 and 16.

```
public class Solid
{
    private String name;

    //constructor
    public Solid(String solidName)
    { name = solidName; }

    public String getName()
    { return name; }

    public double volume()
    { /* implementation not shown */ }
}

public class Sphere extends Solid
{
    private double radius;

    //constructor
    public Sphere(String sphereName, double sphereRadius)
    {
        super(sphereName);
        radius = sphereRadius;
    }

    public double volume()
    { return (4.0/3.0) * Math.PI * radius * radius * radius; }
}

public class RectangularPrism extends Solid
{
    private double length;
    private double width;
    private double height;

    //constructor
    public RectangularPrism(String prismName, double l, double w,
            double h)
    {
        super(prismName);
        length = l;
        width = w;
        height = h;
    }

    public double volume()
    { return length * width * height; }
}
```

15. A program that tests these classes has the following declarations and assignments:

```
Solid s1, s2, s3, s4;
s1 = new Solid("blob");
s2 = new Sphere("sphere", 3.8);
s3 = new RectangularPrism("box", 2, 4, 6.5);
s4 = null;
```

How many of the above lines of code are incorrect?

(A) 0
(B) 1
(C) 2
(D) 3
(E) 4

16. Here is a program that prints the volume of a solid:

```
public class SolidMain
{
    /** Output volume of Solid s. */
    public static void printVolume(Solid s)
    {
        System.out.println("Volume = " + s.volume() +
                " cubic units");
    }

    public static void main(String[] args)
    {
        Solid sol;
        Solid sph = new Sphere("sphere", 4);
        Solid rec = new RectangularPrism("box", 3, 6, 9);
        int flipCoin = (int) (Math.random() * 2);    //0 or 1
        if (flipCoin == 0)
            sol = sph;
        else
            sol = rec;
        printVolume(sol);
    }
}
```

Which is a true statement about this program?

(A) It will output the volume of the sphere or box, as intended.
(B) It will output the volume of the default `Solid s`, which is neither a sphere nor a box.
(C) It will randomly print the volume of sphere or a box.
(D) A run-time error will occur because it is not specified whether `s` is a sphere or a box.
(E) A run-time error will occur because of parameter type mismatch in the method call `printVolume(sol)`.

17. Consider these class declarations.

```
public class Player
{
    public Player()
    { /* implementation not shown */ }

    public int getMove()
    { /* implementation not shown */ }

    //Other constructors and methods not shown.
}

public class ExpertPlayer extends Player
{
    public int compareTo(ExpertPlayer expert)
    { /* implementation not shown */ }

    //Constructors and other methods not shown.
}
```

Which code segment in a client program will cause an error?

```
I   Player p1 = new ExpertPlayer();
    int x1 = p1.getMove();

II  int x;
    ExpertPlayer c1 = new ExpertPlayer();
    ExpertPlayer c2 = new ExpertPlayer();
    if (c1.compareTo(c2) < 0)
        x = c1.getMove();
    else
        x = c2.getMove();

III int x;
    Player h1 = new ExpertPlayer();
    Player h2 = new ExpertPlayer();
    if (h1.compareTo(h2) < 0)
        x = h1.getMove();
    else
        x = h2.getMove();
```

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

18. Consider the following class definitions.

```
public class Animal
{
    private String type;

    public Animal(String theType)
    {
        type = theType;
    }

    public String getType()
    {
        return type;
    }
}

public class Dog extends Animal
{
    public Dog(String theType)
    {
        super(theType);
    }
}
```

The following code segment appears in a class other than `Animal` or `Dog`.

```
Animal d1 = new Animal("poodle");
Animal d2 = new Dog("shnauzer");
Dog d3 = new Dog("yorkie");

public static void display(Animal a)
{
    System.out.println("This dog is a " + a.getType();)
}
```

Which of the following method calls will compile without error?

I `display(d1);`

II `display(d2);`

III `display(d3);`

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

19. Consider the following class definitions.

```
public class StrStuff1
{
    public void printSub(String str)
    {
        String s = str.substring(2);
        System.out.print(s);
    }
}

public class StrStuff2 extends StrStuff1
{
    public void printSub(String str)
    {
        String s = str.substring(1);
        super.printSub(s);
        System.out.print(s);
    }
}
```

The following code segment appears in a class other than `StrStuff1` and `StrStuff2`.

```
    StrStuff1 p = new StrStuff2();
    p.printSub("crab");
```

What is printed as a result of executing the code segment?

(A) `crabab`
(B) `brab`
(C) `rabb`
(D) `abb`
(E) `ab`

20. Consider the following class definitions.

```
public class Class1
{
    public void doSomething(int n)
    {
        n -= 4;
        System.out.print(n);
    }
}

public class Class2 extends Class1
{
    public void doSomething(int n)
    {
        super.doSomething(n + 3);
        n *= 2;
        System.out.print(n);
    }
}
```

The following code segment appears in a class other than `Class1` and `Class2`.

```
Class1 c = new Class2();
c.doSomething(8);
```

What is printed as a result of executing the code segment?

(A) 416
(B) 422
(C) 714
(D) 716
(E) 722