

CSA2

1.Consider the following class declarations.

public class Publication

{

private String title;

public Publication()

{

title = "Generic";

}

public Publication(String t)

{

title = t;

}

}

public class Book extends Publication

{

public Book()

{

super();

}

public Book(String t)

{

super(t);

}

}

The following code segment appears in a method in another class.

Book myBook = new Book("Adventure Story"); // Line 1

Book yourBook = new Book(); // Line 2

Which of the following best describes the result of executing the code segment?

(A) Object myBook is created using the one-argument Book constructor, which uses super to set myBook’s title attribute to "Adventure Story". Object yourBook is created using the Book constructor, which uses super to set yourBook’s title attribute to an empty string.

(B) Object myBook is created using the no-argument Book constructor, which uses super to set myBook’s title attribute to "Generic". Object yourBook is created using super to call to the Publication no-argument constructor to set yourBook’s title attribute to "Generic".

(C) Object myBook is created using the one-argument Book constructor, which uses super to set myBook’s title attribute to "Adventure Story". Object yourBook is created using super to call to the Publication no-argument constructor to set yourBook’s title attribute to "Generic".

(D) A runtime error occurs in line 1 because the one-argument Publication constructor cannot be called from the one-argument Book constructor.

(E) A runtime error occurs in line 2 because the no-argument Publication constructor cannot be called from the no-argument Book constructor.

2.Consider the following class declarations.

public class ParentClass

{

public void wheelsOnTheBus()

{

System.out.println("round and round");

}

}

public class SubClass extends ParentClass

{

public void wheelsOnTheBus()

{

System.out.println("are flat");

}

}

public class SubSubClass extends ParentClass

{

public void wheelsOnTheBus()

{

// No methods defined

}

}

The following code segment appears in a method in another class.

obj.wheelsOnTheBus();

Under which of the following conditions will the code segment print "are flat" ?

I. when obj has been declared as type ParentClass

II. when obj has been declared as type SubClass

III. when obj has been declared as type SubSubClass

(A) I only

(B) II only

(C) I and II only

(D) II and III only

(E) I, II, and III

3.Consider the following class definitions.

public class Appliance

{

private int id;

private String brand;

public Appliance(int aId, String aBrand)

{ /\* implementation not shown \*/ }

public String display()

{ /\* implementation not shown \*/ }

}

public class Refrigerator extends Appliance

{

private int numOfDoors;

public Refrigerator(int rId, String rBrand, int rNumOfDoors)

{ /\* implementation not shown \*/ }

}

The following code segment appears in a class other than Appliance or Refrigerator.

public static void displayFeatures(Refrigerator r)

{

System.out.println(r.display()); // Line 3

}

Appliance a1 = new Refrigerator(456, "AllBrand", 2); // Line 6

Refrigerator a2 = new Refrigerator(789, "Xtreme", 3); // Line 7

displayFeatures(a1); // Line 8

displayFeatures(a2); // Line 9

Which of the following best explains why the code segment will not compile?

(A) Line 3 causes a compile-time error because the Refrigerator class is missing the display() method.

(B) Line 6 causes a compile-time error because the variable a1 is incorrectly instantiated.

(C) Line 7 causes a compile-time error because the variable a2 is incorrectly instantiated.

(D) Line 8 causes a compile-time error because the parameter a1 in the call displayFeatures(a1) has the incorrect data type.

(E) Line 9 causes a compile-time error because the parameter a2 in the call displayFeatures(a2) has the incorrect data type.

4.Consider the following class definitions.

public class Aclass

{

public void methodX()

{

System.out.print("Super X ");

methodY();

}

public void methodY()

{

System.out.print("Super Y ");

methodZ();

}

public void methodZ()

{

System.out.print("Super Z");

}

}

public class Bclass extends Aclass

{

public void methodX()

{

super.methodX();

}

public void methodY()

{

System.out.print("Sub Y ");

methodZ();

}

}

The following code segment appears in a class other than Aclass or Bclass.

Aclass thing = new Bclass();

thing.methodX();

The code segment is intended to display the following.

Super X Super Y Super Z

Which of the following best explains why the code segment does not work as intended?

(A) The variable thing should be declared as a Bclass data type because thing is instantiated as a Bclass object.

(B) The variable thing should be instantiated as an Aclass object because methodY is overridden in Bclass.

(C) The method methodX should be removed from the Aclass definition because methodX is overridden in Bclass.

(D) The method methodY should be removed from the Aclass definition because methodY is overridden in Bclass.

(E) The method methodZ should be overridden in the Bclass definition because methodZ appears only in Aclass.

5.Consider the following class definition.

public class Time

{

private int hours;

private int minutes;

public Time(int h, int m)

{

hours = h;

minutes = m;

}

public boolean equals(Object other)

{

if (other == null)

{

return false;

}

Time t = (Time) other;

return (hours \* 60 + minutes == t.hours \* 60 + t.minutes);

}

}

The following code segment appears in a class other than Time.

Time t1 = new Time(1, 10);

Time t2 = new Time(0, 70);

Which of the following statements will print true ?

I. System.out.println(t1 == t2);

II. System.out.println(t1.equals(t2));

III. System.out.println(equals(t1, t2);

(A) I only

(B) II only

(C) III only

(D) I and II

(E) I and III

Questions 6–8 refer to the BankAccount, SavingsAccount, and CheckingAccountclasses defifined 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 \*/ }

}

6.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

7.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

8.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;

9.FRQ

SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

• Assume that the classes listed in the Java Quick Reference have been imported where appropriate.

• Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.

• In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

The following Book class is used to represent books and print information about each book. Each Book object has attributes for the book title and for the name of the book’s author.

public class Book

{

private String title;

private String author;

public Book(String t, String a)

{

title = t;

author = a;

}

public void printBookInfo()

{

System.out.print(title + ", written by " + author);

}

}

1. The PictureBook class is a subclass of the Book class that has one additional attribute: a String variable named illustrator that is used to represent the name of the illustrator of a picture book. The PictureBook class also contains a printBookInfo method to print the title, writer, and illustrator of a picture book.

Consider the following code segment.

PictureBook myBook = new PictureBook("Peter and Wendy", "J.M. Barrie",

"F.D. Bedford");

myBook.printBookInfo();

The code segment is intended to print the following output.

Peter and Wendy, written by J.M. Barrie and illustrated by F.D. Bedford

Complete the PictureBook class below. Your implementation should conform to the example above.

public class PictureBook extends Book

Consider the following books.

A book titled *Frankenstein*, written by Mary Shelley

A picture book titled *The Wonderful Wizard of Oz*, written by L. Frank Baum and illustrated by W.W. Denslow

The following code segment is intended to represent the two books described above as objects book1 and book2, respectively, and add them to the ArrayList myLibrary.

ArrayList<Book> myLibrary = new ArrayList<Book>();

/\* missing code \*/

myLibrary.add(book1);

myLibrary.add(book2);

1. Write a code segment that can be used to replace /\* missing code \*/ so that book1 and book2 will be correctly created and added to myLibrary. Assume that class PictureBook works as intended, regardless of what you wrote in part (a).

The BookListing class is used to generate a descriptive listing for a book. The BookListing constructor takes a

Book object and a double value as parameters and uses them to print information about the book, along with its price.

Assume that book1 and book2 were created as specified in part (b). The following table demonstrates the intended

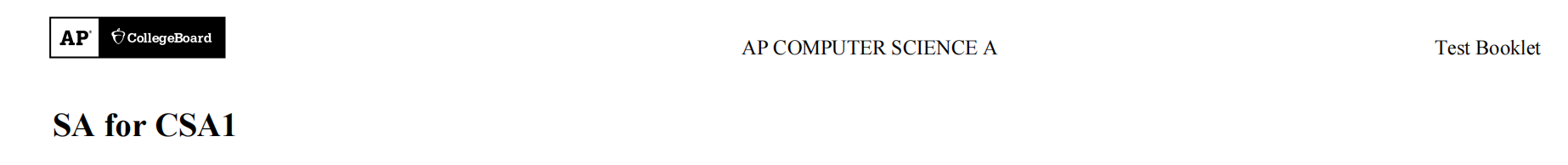
behavior of the BookListing class using objects book1 and book2.

|  |  |
| --- | --- |
| **Code Segment** | **Result Printed** |
| BookListing listing1 = new BookListing(book1, 10.99);listing1.printDescription(); | Frankenstein, written by Mary Shelley, $10.99 |
| BookListing listing2 = new BookListing(book2, 12.99);listing2.printDescription(); | The Wonderful Wizard of Oz, written by L. Frank Baum and illustrated by W.W. Denslow, $12.99 |

(c)  Complete the BookListing class below. Your implementation should conform to the examples. Assume that

class PictureBook works as intended, regardless of what you wrote in part (a).

public class BookListing



CSA2

MCQ (2points for each)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  |  |  |  |  |  |  |

FRQ (4points+2points+3points)