Q1. How many instance variables are created in a SavingsAccount object?

**2**

Q2. Can the method m() of the SavingsAccount directly access the balance directly as shown below? Yes or No.

public void m()

{

System.out.println(balance);

}

**No**

Q3. What happens if the SavingsAccount defines another instance variable named "balance"? Compilation error or Logic error?

**Logic error**

**Subclass Constructors**

Q4. What does Java do if a class does not define any explicit constructor?

**It calls the no args/ default constructor of the superclass**

Q5. What does super(...) do when you call it in the constructor of a subclass?

**Initializes the variables that were inherited from the immediate superclass**

Q6. Can you call super(...) in anywhere in the constructor of a subclass?

**No, super comes first**

Q7. What does Java do if you don't explicitly call super(...) in the constructor of a subclass?

**Compiler ads super(); //Calls no args constructor of super**

**Subclass Methods**

Q8. Write three different types of methods available for a subclass object.

**Inherited, Overridden, and newly defined**

Q9. What does super.method\_name(....) do when you call in the overriden method of a subclass?

**Calls the same method of closest superclass**

**Precondition and Postcondition of Overriden methods.**

Q10. Explain what is a precondition.

**A precondition is a condition that must be true when method is called**

Q11. Explain what is a postcondition.

**Condition that must be true after method is done executing**

Q12. Write a simple method to show how the method enforces a predoncition.

**/\*\***

**@precondition x > 0**

**\*/**

**Public int divide100byX(int x){**

**If(x == 0) throw new IllegalArgumentException();**

**Return 100 / x;**

**}**

Q13. Consider the rule associated with the precondition of an overriden method:"Precondition of overriden method cannot be stronger than that of its inherted method." Give an example that violates this rule and explain how this violation hinders the code reusability. You may use the same example I explain in the tutorial.

A subclass method can only require a *precondition that is at most as strong as* the precondition of the inherited method that it overrides.

public class Employee

{ /\*\*

Sets the employee salary to a given value.

@param aSalary the new salary

@precondition aSalary > 0

\*/ public void setSalary(double aSalary) { ... }}

public class Manager extends Employee

/\*\*

precondition: Salary > 100,000 / stronger precondition

/\*

public void setSalary (double aSalary){

if (aSalary <= 100000) throw an exception;

salary = Salary + bonus;

}

**The reason the precondition can’t be stronger is because it hinders inheritance. If you try to do this**

public void someMethod(Employee e)

{ e.setSalary(50000); }

**An error will be thrown**

**Multiple Inheritance**

Q14. Why doesn't Java support multiple inheritance**? Because it causes an ambiguity problem**

Q15. How do you express multiple IS-a relationship in Java? **With interfaces. …….Implements interface1, interface2**