Sahil A. Nawab – Classes Exercises

AntPopulation.java

```
package Classes;

public class AntPopulation {
	// INSTANCE VARIABLES
		private int population, breedTime;

	// CONSTRUCTORS
		/**
		 * Creates an AntPopulation object with the initial population
		 * and sets breeding time to specified amount in days (how many
		 * days does it take to double the population)
		 * @param initialPop
		 */
		public AntPopulation(int initialPop, int time) {
			population = initialPop;
			breedTime = time;
		} // end constructor

	// MEMBER METHODS
		/**
		 * Breeds the ant population for specified amount of time based
		 * on breeding rate (ie. if breeding rate is 1 day, it takes
		 * day for the population to double and if the breeding time is
		 * set to 2 days, the population will quadruple)
		 * @param days
		 */
		public void breed(int days) {
			population += population * (Math.pow(2, breedTime));
		} // end method breed

		/**
		 * Sprays insecticide and kills 10 percent of the population
		 */
		public void spray() {
			population -= population * 0.1;
		} // end method spray

		/**
		 * Returns current ant population
		 * @return
		 */
		public int getAnts() {
			return population;
		} // end method getAntPopulation

} // end class AntPopulation
```

AntPopulationTester.java

```
package Classes;
```

```
public class AntPopulationTester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		AntPopulation ants = new AntPopulation(100, 1);

		ants.breed(5);
		ants.spray();

		System.out.println(ants.getAnts());
	} // end main method

} // end class AntPopulationTester
```

BankAccount.java

```
package Classes;

public class BankAccount {

	// INSTANCE VARIABLES
		private int accountNum;
		private double balance;

	// CONSTRUCTORS
		public BankAccount(int num, double bal) {
			accountNum = num;
			balance = bal;

			System.out.println("Account Number: " + accountNum);
			System.out.println("Balance: " + balance);
		} // end constructor

		public BankAccount() {
			accountNum = 0;
			balance = 0;

			System.out.println("All fields initialized to 0");
		} // end constructor

	// MEMBER METHODS

	/**
	 * Deposits specified amount into account
	 * @param amount
	 */
	public void deposit(double amount) {
		balance += amount;
	} // end method deposit

	/**
	 * Withdraws specified amount from account
	 * @param amount
	 */
	public void withdraw(double amount) {
		balance -= amount;
	} // end method withdraw
```

```
	/**
	 * Adds interest at specified rate
	 * @param rate
	 */
	public void addInterest(double rate) {
		double interest = balance * rate;
		balance += interest;

		System.out.println("Rate: " + rate * 100 + "%");
		System.out.println("Balance: " + getBalance());
	} // end method addInterest

	// GETTERS AND SETTERS

	/**
	 * Returns current balance
	 * @return
	 */
	public double getBalance() {
		return balance;
	} // end method getBalance

} // end class BankAccount
```

BankAccountTester.java

```
package Classes;

public class BankAccountTester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		BankAccount account = new BankAccount();

		account.deposit(1000);
		account.withdraw(500);
		account.withdraw(400);

		System.out.println(account.getBalance());
	} // end main method

} // end class BankAccountTester
```

Bug.java

```
package Classes;

public class Bug {
	// INSTANCE VARIABLES
		private int position;
		private boolean direction; // true is right, false is left

	// CONSTRUCTORS
		/**
		 * Constructs a Bug object with an initial position
		 * @param initialPosition
```

```
	 */
	public Bug(int initialPosition) {
		position = initialPosition;
	} // end constructor

// MEMBER METHODS
	/**
	 * Changes direction from true (go right) to false (go left) or
	 * from false (go left) to true (go right)
	 */
	public void turn() {
		direction = !direction;
	} // end method turn

	/**
	 * Moves bug by one unit to the right if direction is true and
	 * one unit to the left if direction is false
	 */
	public void move() {
		if (direction) {
			position++;
		}
		else {
			position--;
		}
	} // end method move

	/**
	 * Returns current position of bug
	 * @return
	 */
	public int getPosition() {
		return position;
	} // end method getPosition
} // end class Bug
```

BugTester.java

```
package Classes;

public class BugTester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Bug bug = new Bug(100);

		bug.move();
		bug.move();
		bug.move();

		bug.turn();
		bug.move();

		System.out.println(bug.getPosition());
	} // end main method

} // end class BugTester
```

Car.java

```
package Classes;

public class Car {
	// INSTANCE VARIABLES
	private int mpg;
	private double fuel;

	// CONSTRUCTORS
	/**
	 * Constructs a Car object with fuel efficiency in miles per gallon
	 * @param fuelEfficiency
	 */
	public Car(int fuelEfficiency) {
		mpg = fuelEfficiency;
	} // end constructor

	// MEMBER METHODS
	/**
	 * "Drives" the car the specified distance and "uses" the correct
	 * amount of gas based on the car's mileage
	 * @param distance
	 */
	public void drive(int distance) {
		double fuelUsed = (double) distance / (double) mpg;
		fuel -= fuelUsed;
	} // end method drive

	/**
	 * Adds specified amount of gas (in gallons) to the fuel tank
	 * @param gas
	 */
	public void addGas(double gas) {
		fuel += gas;
	} // end method addGas

	/**
	 * Returns the current amount of gas in the fuel tank
	 * @return
	 */
	public double getGasInTank() {
		return fuel;
	} // end method getGasInTank

} // end class Car
```

CarTester.java

```
package Classes;

public class CarTester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Car myBMW = new Car(22); // 22 miles per gallon
```

```
            myBMW.addGas(28); // 28 gallons of gas
            myBMW.drive(100); // 100 miles driven

            System.out.println(myBMW.getGasInTank()); // Gas remaining
      } // end main method

} // end class CarTester
```

CashRegister.java

```
package Classes;

import java.util.ArrayList;

public class CashRegister {

      // INSTANCE VARIABLES
            private double total;
            private double subtotal;
            private double payment;
            private double change;

            private double tax;
            private final double TAX_RATE;

            private ArrayList<Item> items = new ArrayList<Item>();

      // CONSTRUCTORS
            /**
             * Creates a CashRegister object with a tax rate in decimal form
             * (ie. 5% would be entered as 0.05)
             * @param taxRate
             */
            public CashRegister(double taxRate) {
                  total = 0;
                  payment = 0;
                  change = 0;

                  tax = 0;
                  TAX_RATE = taxRate;
            } // end constructor

      // MEMBER METHODS
            /**
             * Adds item using an item class
             * @param item
             */
            public void addItem(Item item) {
                  items.add(item);
                  subtotal += item.getItemPrice();
            } // end method addItem

            /**
             * Adds item using inputs as name and price and automatically
             * instantiates an Item using those parameters
             * @param n
             * @param p
```

```java
 */
public void addItem(String n, double p) {
	Item item = new Item(n, p);
	addItem(item);
} // end method addItem

/**
 * Adds customer payment
 * @param input
 */
public void addPayment(double input) {
	payment = input;
} // end method customerPayment

/**
 * Calculates tax based on initial tax rate and adds tax to total
 */
public void calculateTax() {
	tax = subtotal * TAX_RATE;
	total = subtotal + tax;
} // end method calculateTax

/**
 * Calculates change based on total (after tax)
 */
public void calculateChange() {
	change = payment - total;
} // end method calculateChange

/**
 * Prints a receipt
 */
public void printReceipt() {
	System.out.println("----- Customer Receipt -----");
	System.out.println(); // blank line for spacing

	for (int i = 0; i < items.size(); i++) {
		System.out.println("Item " + (i + 1) + ": " +
			items.get(i).getItemName() + " | $" +
			items.get(i).getItemPrice());
	}

	System.out.println(); // blank line for spacing

	System.out.println("Subtotal: $" + subtotal);
	calculateTax();
	System.out.println("Tax: $" + tax);
	System.out.println("Total: $" + total);

	System.out.println(); // blank line for spacing

	System.out.println("Customer Payment: $" + payment);
	calculateChange();
	System.out.println("Change: $" + change);
		// later implement coin calculation

} // end method printReceipt
```

```
	/**
	 * Returns sales total
	 * @return
	 */
	public double getSalesTotal() {
		return subtotal;
	} // end method getSalesTotal

	/**
	 * Returns number of sales
	 * @return
	 */
	public int getSalesCount() {
		return items.size();
	} // end method getSalesCount

	/**
	 * Resets all values to 0 except tax rate
	 */
	public void reset() {
		total = 0;
		payment = 0;
		change = 0;
		tax = 0;
	} // end method reset

} // end class CashRegister
```

CashRegisterTester.java

```
package Classes;

import java.util.Scanner;

public class CashRegisterTester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner keyboard = new Scanner(System.in);
		CashRegister register = new CashRegister(0.0625);

		boolean cont;

		String name;
		double price;

		// get user input
		do {
			cont = false;

			System.out.print("Please enter name of item: ");
			name = keyboard.nextLine();

			System.out.print("Please enter price of item: ");
			price = keyboard.nextDouble();

			register.addItem(name, price);
```

```
                keyboard.nextLine(); // absorb extra newlines
                System.out.print("Add another item? (Y/N): ");
                if (keyboard.nextLine().equalsIgnoreCase("y")) {
                      cont = true;
                }

                // resets
                name = "";
                price = 0;
          } while (cont);

          System.out.print("Please enter customer payment: ");
          register.addPayment(keyboard.nextDouble());

          System.out.println(); // blank line for spacing
          register.printReceipt();
    } // end main method

} // end class CashRegisterTester
```

Employee.java

```
package Classes;

public class Employee {

    // INSTANCE VARIABLES
          private String name;
          private double salary;

    // CONSTRUCTORS
          public Employee(String employeeName, double currentSalary) {
                name = employeeName;
                salary = currentSalary;
          } // end constructor

    // MEMBER METHODS
    public String getName() {
          return name;
    } // end method getName

    public double getSalary() {
          return salary;
    } // end method getSalary

    public void raiseSalary(double percent) {
          salary += salary * (percent / 100.0);
    } // end method raiseSalary

} // end class Employee
```

EmployeeTester.java

```
package Classes;

public class EmployeeTester {
```

```
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Employee emp1 = new Employee("Shelly", 100);

		System.out.println(emp1.getName());
		System.out.println(emp1.getSalary());
		emp1.raiseSalary(10);

		System.out.println(emp1.getSalary());
	} // end main method

} // end class EmployeeTester
```

Item.java

```
package Classes;

public class Item {

	// INSTANCE VARIABLES
	private String name;
	private double price;
	private int quantity;

	// CONSTRUCTORS

	/**
	 * Creates an Item object with a name, price, and quantity
	 * @param n
	 * @param p
	 * @param q
	 */
	public Item(String n, double p, int q) {
		name = n;
		price = p;
		quantity = q;
	} // end constructor

	/**
	 * Creates an Item object with a name and price
	 * Quantity is set to 1
	 * @param n
	 * @param p
	 */
	public Item(String n, double p) {
		name = n;
		price = p;
		quantity = 1;
	} // end constructor

	// MEMBER METHODS

	/**
	 * Returns name of item
	 * @return
	 */
```

```
	public String getItemName() {
		return name;
	} // end method getName

	/**
	 * Returns price of item
	 * @return
	 */
	public double getItemPrice() {
		return price;
	} // end method getPrice

	/**
	 * Returns quantity of item
	 * @return
	 */
	public int getItemQuantity() {
		return quantity;
	} // end method getItemQuantity

} // end class Item
```

Rational.java

```
package Classes;

/**
 * The Rational object contains a fraction with numerator and denominator
 * along with fractional operators (multiply, divide, add, and subtract)
 * as well as simplification and printing
 * @author Sahil
 *
 */
public class Rational {

	// INSTANCE VARIABLES
		private int num, den;
		private int numOrig, denOrig;

	// CONSTRUCTORS
		/**
		 * Creates a Rational object with a numerator and denominator
		 * Keeps original inputs separate from reduced values
		 * @param numerator
		 * @param denominator
		 */
		public Rational(int numerator, int denominator) {
			numOrig = numerator;
			denOrig = denominator;

			reduce();
		} // end constructor

	// MEMBER METHODS
		/**
		 * Allows original numerator to be changed and reduces fraction
		 * @param numerator
```

```java
 */
public void replaceNumerator(int numerator) {
      numOrig = numerator;
      reduce();
} // end method replaceNumerator

/**
 * Allows original denominator to be changed and reduces fraction
 * @param denominator
 */
public void replaceDenominator(int denominator) {
      numOrig = denominator;
      reduce();
} // end method replaceDenominator

/**
 * Allows both original numerator and denominator to be changed
 * and reduces fraction
 * @param numerator
 * @param denominator
 */
public void replaceBoth(int numerator, int denominator) {
      numOrig = numerator;
      denOrig = denominator;
} // end method replaceBoth

/**
 * Returns original numerator
 * @return
 */
public int getOriginalNumerator() {
      return numOrig;
} // end method getOriginalNumerator

/**
 * Returns original denominator
 * @return
 */
public int getOriginalDenominator() {
      return denOrig;
} // end method getOriginalDenominator

/**
 * Returns reduced numerator
 * @return
 */
public int getNumerator() {
      return num;
} // end method getNumerator

/**
 * Returns reduced denominator
 * @return
 */
public int getDenominator() {
       return den;
} // end method getDenominator
```

```java
/**
 * Returns decimal approximation
 * @return
 */
public double getDecimal() {
      return (double) num / (double) den;
} // end method getDecimal

/* Less efficient algorithm?
public int gcf(int a, int b) {
      while (a != 0 && b != 0) {
    if (a >= b) {
        a = a - b;
    }
    else {
      b = b - a;
    }
}
if (a == 0) return b;
else return a;
} // end method gcf
*/

/**
 * Returns greatest common factor of specified values
 * @param a
 * @param b
 * @return
 */
public int gcf(int a, int b) {
      return b == 0 ? a : gcf(b, a % b);
} // end method gcf

/**
 * Reduces/simplifies fraction
 */
public void reduce() {
      int gcf = gcf(num, den);
      num = num / gcf;
      den = den / gcf;
} // end method reduce

/**
 * Returns Rational object as a String (kind of like toString)
 * @return
 */
public String getRational() {
      boolean negative = false;

      if (num < 0) {
            negative = true;
            num = 0 - num;
      }
      if (den < 0) {
            negative = true;
            den = 0 - den;
```

```
        }

        if (negative) {
                return "(-)" + num + "/" + den;
        }
        else {
                return num + "/" + den;
        }
} // end method getRational

/**
 * Returns original fraction as a string
 * @return
 */
public String getOriginal() {
        return numOrig + "/" + denOrig;
} // end method getOriginal

        // Fractional operations
        /**
         * Multiplies Rational by specified fraction
         * @param fraction
         * @return
         */
        public Rational multiply(Rational fraction) {
                int frac1Num = num;
                int frac1Den = den;

                int frac2Num = fraction.getNumerator();
                int frac2Den = fraction.getDenominator();

                int resultNum = frac1Num * frac2Num;
                int resultDen = frac1Den * frac2Den;

                Rational result = new Rational(resultNum, resultDen);
                result.reduce();

                return result;
        } // end method multiply

        /**
         * Divides Rational by specified fraction
         * @param fraction
         * @return
         */
        public Rational divide(Rational fraction) {
                int frac1Num = num;
                int frac1Den = den;

                int frac2Num = fraction.getNumerator();
                int frac2Den = fraction.getDenominator();

                int resultNum = frac1Num * frac2Den;
                int resultDen = frac1Den * frac2Num;

                Rational result = new Rational(resultNum, resultDen);
                result.reduce();
```

```
		return result;
	} // end method divide

	/**
	 * Adds Rational by specified fraction
	 * @param fraction
	 * @return
	 */
	public Rational add(Rational fraction) {
		int frac1Num = num;
		int frac1Den = den;

		int frac2Num = fraction.getNumerator();
		int frac2Den = fraction.getDenominator();

		frac1Num = frac1Num * frac2Den;
		frac2Num = frac2Num * frac1Den;

		int resultNum = frac1Num + frac2Num;
		int resultDen = frac1Den * frac2Den;

		Rational result = new Rational(resultNum, resultDen);
		result.reduce();

		return result;
	} // end method add

	/**
	 * Subtracts Rational by specified fraction
	 * @param fraction
	 * @return
	 */
	public Rational subtract(Rational fraction) {
		int frac1Num = num;
		int frac1Den = den;

		int frac2Num = fraction.getNumerator();
		int frac2Den = fraction.getDenominator();

		frac1Num = frac1Num * frac2Den;
		frac2Num = frac2Num * frac1Den;

		int resultNum = frac1Num - frac2Num;
		int resultDen = frac1Den * frac2Den;

		Rational result = new Rational(resultNum, resultDen);
		result.reduce();

		return result;
	} // end method subtract

} // end class Rational
```

RationalTester.java

```
package Classes;
```

```
import java.util.Scanner;

public class RationalTester {
	public static void main (String args[]) {

		Scanner keyboard = new Scanner(System.in);
		System.out.print("Enter the first numerator ----> ");
		int num1 = keyboard.nextInt();
		System.out.print("Enter the first denominator --> ");
		int den1 = keyboard.nextInt();
		System.out.print("Enter the second numerator ----> ");
		int num2 = keyboard.nextInt();
		System.out.print("Enter the second denominator --> ");
		int den2 = keyboard.nextInt();

		Rational fraction1 = new Rational(num1, den1);
		Rational fraction2 = new Rational(num2, den2);

		String frac1Orig = fraction1.getOriginal();
		String frac2Orig = fraction2.getOriginal();

		System.out.println(); // blank line for spacing

		// display fraction1 information
			System.out.println(frac1Orig + " approximates to " +
fraction1.getDecimal());
			if (frac1Orig.equals(fraction1.getRational())) {
				System.out.println("and is already in simplest
form");
			}
			else {
				System.out.println("and reduces to " +
fraction1.getRational());
			}

		// display fraction2 information
			System.out.println(frac2Orig + " approximates to " +
fraction2.getDecimal());
			if (frac2Orig.equals(fraction2.getRational())) {
				System.out.println("and is already in simplest
form");
			}
			else {
				System.out.println("and reduces to " +
fraction2.getRational());
			}

		System.out.println(); // blank line for spacing

		// display fraction multiplication
			System.out.println(frac1Orig + " * " + frac2Orig + " equals
" + fraction1.multiply(fraction2).getRational());
			System.out.println(frac1Orig + " / " + frac2Orig + " equals
" + fraction1.divide(fraction2).getRational());
			System.out.println(frac1Orig + " + " + frac2Orig + " equals
" + fraction1.add(fraction2).getRational());
```

```
                  System.out.println(frac1Orig + " - " + frac2Orig + " equals
" + fraction1.subtract(fraction2).getRational());
      }
} // end class RationalTester
```

SavingsAccount.java

```
package Classes;

public class SavingsAccount {

      // INSTANCE VARIABLES
            private double balance;
            private final double INTEREST_RATE;

      // CONSTRUCTORS
            public SavingsAccount(double initialBal, double interestRate) {
                  balance = initialBal;
                  INTEREST_RATE = interestRate;
            } // end constructor

      // MEMBER METHODS
            /**
             * Deposits specified amount into account
             * @param amount
             */
            public void deposit(double amount) {
                  balance += amount;
            } // end method deposit

            /**
             * Withdraws specified amount from account
             * @param amount
             */
            public void withdraw(double amount) {
                  balance -= amount;
            } // end method withdraw

            /**
             * Adds interest at given rate
             */
            public void addInterest() {
                  double interest = balance * INTEREST_RATE;
                  balance += interest;

                  System.out.println("Rate: " + INTEREST_RATE * 100 + "%");
                  System.out.println("Balance: " + getBalance());
            } // end method addInterest

            /**
             * Returns current balance
             * @return
             */
            public double getBalance() {
                  return balance;
            } // end method getBalance
```

```
} // end class SavingsAccount
```

SavingsAccountTester.java

```
package Classes;

public class SavingsAccountTester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		SavingsAccount account = new SavingsAccount(100, 0.0625);

		account.deposit(1000);
		account.withdraw(500);
		account.withdraw(400);

		System.out.println("Balance (prior): " + account.getBalance());

		account.addInterest();

		System.out.println("Balance (post): " + account.getBalance());
	} // end main method

} // end class SavingsAccountTester
```

Student.java

```
package Classes;

public class Student {
	// INSTANCE VARIABLES
		private String name;
		private int totalScore, numQuiz;

	// CONSTRUCTORS
		/**
		 * Constructs a Student object with a name and sets the number of
		 * quizzes taken to 0 initially
		 * @param n
		 */
		public Student(String n) {
			name = n;
			numQuiz = 0;
		} // end constructor

	// MEMBER METHODS
		/**
		 * Adds a quiz score to total score and logs number of quizzes
		 * taken
		 * @param score
		 */
		public void addQuiz(int score) {
			totalScore += score;
			numQuiz++;
		} // end method addQuiz

		/**
```

```
	 * Returns total score of all quizzes
	 * @return
	 */
	public int getTotalScore() {
		return totalScore;
	} // end method getTotalScore

	/**
	 * Returns average score of all quizzes
	 * @return
	 */
	public double getAverageScore() {
		return totalScore / numQuiz;
	} // end method getAverageScore

} // end class Student
```

StudentTester.java

```
package Classes;

public class StudentTester {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Student myStudent = new Student("Charles Griffin");

		myStudent.addQuiz(90);
		myStudent.addQuiz(94);
		myStudent.addQuiz(96);

		System.out.println(myStudent.getTotalScore());
		System.out.println(myStudent.getAverageScore());
	} // end main method

} // end class StudentTester
```