**Q1:**

public class Account {

private double balance;

public final double MIN\_BALANCE;

public Account(double initialBalance, double minBalance) {

MIN\_BALANCE = minBalance;

balance = initialBalance;

}

public double withdraw(double amount) throws InsufficientFundsException {

if (amount > balance) {

throw new InsufficientFundsException();

} else {

balance -= amount;

System.out.println("Withdrawal successful");

}

return balance;

}

public static void main(String[] args) {

Account account = new Account(1000, 0);

account.withdraw(1500);

}

}

class InsufficientFundsException extends Exception {

public InsufficientFundsException() {

super("Insufficient funds for the withdrawal");

}

}

**Question**: Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation**: The program will have a compilation error because the InsufficientFundsException is not catch within the main method.

**Mark**: One mark for error, one mark for explanation

**Q2:**

public class Account {

private double balance;

public final double MIN\_BALANCE;

public Account(double initialBalance, double minBalance) {

MIN\_BALANCE = minBalance;

balance = initialBalance;

}

public double withdraw(double amount) throws InsufficientFundsException {

if (amount > balance) {

throw new InsufficientFundsException();

} else {

balance -= amount;

System.out.println("Withdrawal successful");

}

return balance;

}

public static void main(String[] args) {

Account account = new Account(1000, 0);

try {

account.withdraw(1500);

} catch (InsufficientFundsException e) {

System.out.println("Failed to withdraw: " + e.getMessage());

}

}

}

class InsufficientFundsException extends Exception {

public InsufficientFundsException() {

super("Insufficient funds for the withdrawal");

}

}

**Question:** Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program will throw an InsufficientFundsException because the withdrawal amount (1500) is greater than the current balance (1000). However, this time, the exception is caught within a try-catch block, so the program will handle the exception and print a custom error message “Failed to withdraw: Insufficient funds for the withdrawal”.

**Mark**: 1 mark for no error, 1 mark for explanation

**Q3:**

public class Temperature {

private double celsius;

public Temperature(double initialTemp) {

celsius = initialTemp;

}

public void setCelsius(double temp) throws InvalidTemperatureException {

if (temp < -273.15) {

throw new InvalidTemperatureException();

} else {

celsius = temp;

System.out.println("Temperature set successfully");

}

}

public static void main(String[] args) {

Temperature temp = new Temperature(25.0);

temp.setCelsius(-300);

}

}

class InvalidTemperatureException extends Exception {

public InvalidTemperatureException() {

super("Temperature cannot be below absolute zero (-273.15°C)");

}

}

**Question:** Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program will have a compilation error because the InvalidTemperatureException is not catch within the main method.

Mark: 1 mark for error, 1 mark for explanation

**Q4:**

public class Temperature {

private double celsius;

public Temperature(double initialTemp) {

celsius = initialTemp;

}

public void setCelsius(double temp) throws InvalidTemperatureException {

if (temp < -273.15) {

throw new InvalidTemperatureException();

} else {

celsius = temp;

System.out.println("Temperature set successfully");

}

}

public static void main(String[] args) {

Temperature temp = new Temperature(25.0);

try {

temp.setCelsius(-300);

} catch (InvalidTemperatureException e) {

System.out.println("Failed to set temperature: " + e.getMessage());

}

}

}

class InvalidTemperatureException extends Exception {

public InvalidTemperatureException() {

super("Temperature cannot be below absolute zero (-273.15°C)");

}

}

**Question:** Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program will throw an InvalidTemperatureException because the temperature value -300 is below absolute zero (-273.15°C). However, the exception is caught within a try-catch block, so the program will handle the exception and print a custom error message “Failed to set temperature: Temperature cannot be below absolute zero (-273.15°C)”.

Mark: 1 mark for no error, 1 mark for explanation

Q5:

public class Bank {

private double balance;

public Bank(double initialBalance) {

balance = initialBalance;

}

public void deposit(double amount) throws NegativeDepositException {

if (amount < 0) {

throw new NegativeDepositException();

} else {

balance += amount;

System.out.println("Deposit successful");

}

}

public static void main(String[] args) {

Bank bank = new Bank(1000);

bank.deposit(-500);

}

}

class NegativeDepositException extends Exception {

public NegativeDepositException() {

super("Cannot deposit a negative amount");

}

}

**Question:** Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation**: The program will have a compilation error because the NegativeDepositException is not catch within the main method.

**Mark**: 1 mark for the error, 1 mark for explanation

**Q6**

enum Day {

MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY

}

public class Main {

public static void main(String[] args) {

Day today = Day.WEDNESDAY;

if (today == Day.MONDAY) {

System.out.println("Start of the work week");

} else if (today == Day.WEDNESDAY) {

System.out.println("Midweek");

} else if (today == Day.FRIDAY) {

System.out.println("Almost weekend");

} else {

System.out.println("Weekend!");

}

}

}

**Question:** Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program will output "Midweek" because the today variable is set to Day.WEDNESDAY, and the corresponding if condition matches. The program then prints the associated message and exits the if-else block.

Mark: 1 mark for no error, 1 mark for explanation

**Q7:**

enum TrafficLight {

RED, YELLOW, GREEN

}

public class Main {

public static void main(String[] args) {

for (TrafficLight light : TrafficLight.values()) {

System.out.println(light + " light duration: " + getDuration(light) + " seconds");

}

}

public static int getDuration(TrafficLight light) {

if (light == TrafficLight.RED) {

return 60;

} else if (light == TrafficLight.YELLOW) {

return 5;

} else if (light == TrafficLight.GREEN) {

return 45;

} else {

throw new IllegalArgumentException("Unknown light: " + light);

}

}

}

**Question:** Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program iterates over all values of the TrafficLight enum and prints the duration associated with each light. The output will be:

RED light duration: 60 seconds \nYELLOW light duration: 5 seconds \nGREEN light duration: 45 seconds

Mark: 1 mark for no error, 1 mark for explanation

**Q8:**

enum Season {

WINTER, SPRING, SUMMER, FALL

}

public class Main {

public static void main(String[] args) {

Season currentSeason = Season.SUMMER;

System.out.println("Current season: " + currentSeason);

Season nextSeason = Season.valueOf("FALL");

System.out.println("Next season: " + nextSeason);

Season invalidSeason = Season.valueOf("AUTUMN");

System.out.println("Invalid season: " + invalidSeason);

}

}

**Question:** Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program will throw an IllegalArgumentException at runtime when trying to execute Season.valueOf("AUTUMN") because "AUTUMN" is not a valid constant in the Season enum. The valueOf method expects an exact match with the enum constants, so the program will terminate with an exception before printing "Invalid season".

**Mark:** 1 mark for error, 1 mark for explanation

**Q9:**

enum Level {

LOW(1), MEDIUM(2), HIGH(3);

private int levelCode;

Level(int levelCode) {

this.levelCode = levelCode;

}

public int getLevelCode() {

return levelCode;

}

}

public class Main {

public static void main(String[] args) {

Level myLevel = Level.HIGH;

System.out.println("Level: " + myLevel);

System.out.println("Level code: " + myLevel.getLevelCode());

}

}

**Question**: Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program will output: Level: HIGH\n Level code: 3

The myLevel variable is set to Level.HIGH, and the corresponding getLevelCode() method returns 3, which is the level code associated with HIGH in the enum constructor.

**Mark**: 1 mark for no error, 1 mark for explanation

**Q10：**

enum Planet {

MERCURY("Small and rocky"),

VENUS("Cloudy and hot"),

EARTH("Home to life"),

MARS("The red planet");

private final String description;

Planet(String description) {

this.description = description;

}

public String getDescription() {

return description;

}

}

public class Main {

public static void main(String[] args) {

for (Planet p : Planet.values()) {

System.out.println(p + ": " + p.getDescription());

}

}

}

**Question**: Review the program provided above and identify any errors within it. If you find no errors, state the expected output of the program. Explain your answer.

**Explanation:** The program iterates over all the enum constants of Planet and prints each planet's name along with its description. The output will be:

MERCURY: Small and rocky VENUS: Cloudy and hot EARTH: Home to life MARS: The red planet

**Mark: 1** mark for no error, 1 mark for explanation