**Q1.**

package week\_5;

import java.util.HashMap;

import java.util.Map;

/\*\*

\*

\* Paint color HashMap - keys are room names, values are the color to paint that room.

\*

\* In this program, practice adding data to a HashMap, and analyzing values in a HashMap

\*/

public class Question\_1\_Paint\_Colors {

public static void main(String[] args) {

Map<String, String> paintColors = new HashMap<>();

paintColors.put("Kitchen", "Light Blue");

paintColors.put("Dining Room", "Yellow");

paintColors.put("Living Room", "Yellow");

paintColors.put("Bedroom 1", "Purple");

paintColors.put("Bedroom 2", "Green");

// Add another key-value pair to the map. "Bedroom 3" will be painted "Yellow".

paintColors.put("Bedroom 3", "Yellow");

// Should print "Kitchen color: Light Blue"

System.out.println("Kitchen color: " + getKitchenColor(paintColors));

// Should print "Number of rooms to be painted yellow: 3"

System.out.println("Number of rooms to be painted yellow: " + howManyYellowRooms(paintColors));

}

public static String getKitchenColor(Map<String, String> paintColors) {

// Return the value for the key "Kitchen" in the paintColors HashMap

return paintColors.get("Kitchen");

}

public static int howManyYellowRooms(Map<String, String> paintColors) {

int count = 0;

// Count how many values are the String "Yellow" in the paintColors HashMap

for (String color : paintColors.values()) {

if (color.equals("Yellow")) {

count++;

}

}

return count;

}

}

**Q2.**

package week\_5;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import static input.InputUtils.\*;

/\*\*

\* In Minnesota, we have many lakes. The Minnesota Pollution Control Agency monitors water quality,

\* as part of monitoring Minnesota's environment.

\*

\* One of the measures of lake quality is clarity - how far can you see down into the water?

\* (If you are interested, more info https://www.pca.state.mn.us/water/what-water-clarity-tells-us

\* and https://www.minneapolisparks.org/park\_care\_\_improvements/water\_resources/lake\_water\_resources/)

\*

\* Clarity is measured in distance. A higher distance usually means better water quality.

\*

\* For swimming, going to the beach etc... it's better to have higher water quality.

\* A common recommendation is at least 4 feet or more, for swimming and other water recreation.

\*

\* Finish this program to add and edit lake clarity measurements,

\* and to search for lakes that have the recommended water clarity.

\*/

public class Question\_2\_Lake\_Quality {

public static void main(String[] args) {

Map<String, Double> lakeClarities = new HashMap<>();

lakeClarities.put("Bde Maka Ska", 7.2);

lakeClarities.put("Harriet", 5.3);

lakeClarities.put("Powderhorn", 1.8);

lakeClarities.put("Nokomis", 2.4);

while (yesNoInput("Do you want to add or update a lake clarity?")) {

String lakeName = stringInput("Enter the lake name");

double clarity = positiveDoubleInput("Enter the clarity, in feet");

updateClarity(lakeClarities, lakeName, clarity);

}

// Decide which lakes are suitable for swimming

double minSwimmingClarity = 4.0; // four feet of clarity

List<String> swimmingLakes = getSwimmingLakes(lakeClarities, minSwimmingClarity);

System.out.println("These lakes are suitable for swimming: " + swimmingLakes);

}

public static void updateClarity(Map<String, Double> lakeClarities, String lake, double clarity) {

// TODO add the new lake (key) and clarity (value) to lakeClarities HashMap.

// If the lake name is already in the HashMap, then overwrite the previous clarity value with the new clarity.

// you don't need to return anything. (Make sure you know why?)

lakeClarities.put(lake, clarity);

System.out.println("Updated clarity for " + lake + " to " + clarity + " feet.");

}

public static List<String> getSwimmingLakes(Map<String, Double> lakeClarities, double minClarity) {

// TODO create and return a list of lakes with clarity at or above minClarity.

// Example: if lakeClarities = {"Nokomis"=2.3, "Como"=4.9, "Harriet"=4.0, "Bde Maka Ska"=5.2}

// and minClarity = 4.0

// Return a list of ["Como", "Harriet", "Bde Maka Ska"]

// The order of lakes in this list is not important.

List<String> suitableLakes = new ArrayList<>();

for (String lake : lakeClarities.keySet()) {

double clarity = lakeClarities.get(lake);

if (clarity >= minClarity) {

suitableLakes.add(lake);

}

}

return suitableLakes;

}

**Q3.**

package week\_5;

import java.util.Map;

import static input.InputUtils.stringInput;

/\*\*

\* 1. Create a HashMap of country codes and country names.

\*

\* The 2-letter country codes will be keys

\* The country names will be values.

\*

\* The array countryCodes and countryNames contain (most) of the codes and names for all the countries in the world. The data is a little old and may be missing some countries.

\* The first element in countryCodes (AF) is the code for the first element in countryNames (Afghanistan)

\* The second element in countryCodes (AL) is the code for the second element in countryNames (Aland Islands)

\*

\* Use a loop to add the data to a HashMap.

\*

\* 2. Finish the method that searches for a country name for a country code.

\* This method will search your new HashMap. If the user enters a valid country code, the method will return the country name for that code.

\* If the user enters a code that is not found in the HashMap (is not one of the HashMap keys) your method should return the EXACT String "Code not found"

\*

\*/

public class Question\_3\_Make\_HashMap\_Country {

public static void main(String[] args) {

// You do not need to modify this method.

String[] countryCodes = {"AF", "AX", "AL", "DZ", "AS", "AD", "AO", "AI", "AQ", "AG", "AR", "AM", "AW", "AU", "AT", "AZ", "BS", "BH", "BD", "BB", "BY", "BE", "BZ", "BJ", "BM", "BT", "BO", "BA", "BW",

"BV", "BR", "IO", "BN", "BG", "BF", "BI", "KH", "CM", "CA", "CV", "KY", "CF", "TD", "CL", "CN", "CX", "CC", "CO", "KM", "CG", "CD", "CK", "CR", "CI", "HR", "CU", "CY", "CZ", "DK", "DJ", "DM",

"DO", "EC", "EG", "SV", "GQ", "ER", "EE", "ET", "FK", "FO", "FJ", "FI", "FR", "GF", "PF", "TF", "GA", "GM", "GE", "DE", "GH", "GI", "GR", "GL", "GD", "GP", "GU", "GT", "GG", "GN", "GW", "GY",

"HT", "HM", "VA", "HN", "HK", "HU", "IS", "IN", "ID", "IR", "IQ", "IE", "IM", "IL", "IT", "JM", "JP", "JE", "JO", "KZ", "KE", "KI", "KP", "KR", "KW", "KG", "LA", "LV", "LB", "LS", "LR", "LY",

"LI", "LT", "LU", "MO", "MK", "MG", "MW", "MY", "MV", "ML", "MT", "MH", "MQ", "MR", "MU", "YT", "MX", "FM", "MD", "MC", "MN", "MS", "MA", "MZ", "MM", "NA", "NR", "NP", "NL", "AN", "NC", "NZ",

"NI", "NE", "NG", "NU", "NF", "MP", "NO", "OM", "PK", "PW", "PS", "PA", "PG", "PY", "PE", "PH", "PN", "PL", "PT", "PR", "QA", "RE", "RO", "RU", "RW", "SH", "KN", "LC", "PM", "VC", "WS", "SM",

"ST", "SA", "SN", "CS", "SC", "SL", "SG", "SK", "SI", "SB", "SO", "ZA", "GS", "ES", "LK", "SD", "SR", "SJ", "SZ", "SE", "CH", "SY", "TW", "TJ", "TZ", "TH", "TL", "TG", "TK", "TO", "TT", "TN",

"TR", "TM", "TC", "TV", "UG", "UA", "AE", "GB", "US", "UM", "UY", "UZ", "VU", "VE", "VN", "VG", "VI", "WF", "EH", "YE", "ZM", "ZW"};

String[] countryNames = {"Afghanistan", "Aland Islands", "Albania", "Algeria", "American Samoa", "Andorra", "Angola", "Anguilla", "Antarctica", "Antigua and Barbuda", "Argentina", "Armenia", "Aruba", "Australia",

"Austria", "Azerbaijan", "Bahamas", "Bahrain", "Bangladesh", "Barbados", "Belarus", "Belgium", "Belize", "Benin", "Bermuda", "Bhutan", "Bolivia", "Bosnia and Herzegovina", "Botswana", "Bouvet Island", "Brazil",

"British Indian Ocean Territory", "Brunei Darussalam", "Bulgaria", "Burkina Faso", "Burundi", "Cambodia", "Cameroon", "Canada", "Cape Verde", "Cayman Islands", "Central African Republic", "Chad", "Chile", "China",

"Christmas Island", "Cocos (Keeling) Islands", "Colombia", "Comoros", "Congo", "Congo, The Democratic Republic of the", "Cook Islands", "Costa Rica", "Cote D'Ivoire", "Croatia", "Cuba", "Cyprus", "Czech Republic", "Denmark",

"Djibouti", "Dominica", "Dominican Republic", "Ecuador", "Egypt", "El Salvador", "Equatorial Guinea", "Eritrea", "Estonia", "Ethiopia", "Falkland Islands (Malvinas)", "Faroe Islands", "Fiji", "Finland", "France", "French Guiana",

"French Polynesia", "French Southern Territories", "Gabon", "Gambia", "Georgia", "Germany", "Ghana", "Gibraltar", "Greece", "Greenland", "Grenada", "Guadeloupe", "Guam", "Guatemala", "Guernsey", "Guinea", "Guinea-Bissau", "Guyana",

"Haiti", "Heard Island and Mcdonald Islands", "Holy See (Vatican City State)", "Honduras", "Hong Kong", "Hungary", "Iceland", "India", "Indonesia", "Iran, Islamic Republic Of", "Iraq", "Ireland", "Isle of Man", "Israel", "Italy",

"Jamaica", "Japan", "Jersey", "Jordan", "Kazakhstan", "Kenya", "Kiribati", "Korea, Democratic People\'s Republic of", "Korea, Republic of", "Kuwait", "Kyrgyzstan", "Lao People\'s Democratic Republic", "Latvia", "Lebanon", "Lesotho",

"Liberia", "Libyan Arab Jamahiriya", "Liechtenstein", "Lithuania", "Luxembourg", "Macao", "Macedonia, The Former Yugoslav Republic of", "Madagascar", "Malawi", "Malaysia", "Maldives", "Mali", "Malta", "Marshall Islands", "Martinique", "Mauritania", "Mauritius",

"Mayotte", "Mexico", "Micronesia, Federated States of", "Moldova, Republic of", "Monaco", "Mongolia", "Montserrat", "Morocco", "Mozambique", "Myanmar", "Namibia", "Nauru", "Nepal", "Netherlands", "Netherlands Antilles", "New Caledonia", "New Zealand",

"Nicaragua", "Niger", "Nigeria", "Niue", "Norfolk Island", "Northern Mariana Islands", "Norway", "Oman", "Pakistan", "Palau", "Palestinian Territory, Occupied", "Panama", "Papua New Guinea", "Paraguay", "Peru", "Philippines", "Pitcairn", "Poland", "Portugal",

"Puerto Rico", "Qatar", "Reunion", "Romania", "Russian Federation", "Rwanda", "Saint Helena", "Saint Kitts and Nevis", "Saint Lucia", "Saint Pierre and Miquelon", "Saint Vincent and the Grenadines", "Samoa", "San Marino", "Sao Tome and Principe", "Saudi Arabia", "Senegal",

"Serbia and Montenegro", "Seychelles", "Sierra Leone", "Singapore", "Slovakia", "Slovenia", "Solomon Islands", "Somalia", "South Africa", "South Georgia and the South Sandwich Islands", "Spain", "Sri Lanka", "Sudan", "Suriname", "Svalbard and Jan Mayen",

"Swaziland", "Sweden", "Switzerland", "Syrian Arab Republic", "Taiwan, Province of China", "Tajikistan", "Tanzania, United Republic of", "Thailand", "Timor-Leste", "Togo", "Tokelau", "Tonga", "Trinidad and Tobago", "Tunisia", "Turkey", "Turkmenistan",

"Turks and Caicos Islands", "Tuvalu", "Uganda", "Ukraine", "United Arab Emirates", "United Kingdom", "United States", "United States Minor Outlying Islands", "Uruguay", "Uzbekistan", "Vanuatu", "Venezuela", "Viet Nam", "Virgin Islands, British",

"Virgin Islands, U.S.", "Wallis and Futuna", "Western Sahara", "Yemen", "Zambia", "Zimbabwe"};

Map<String, String> countriesAndCodes = createCountryCodesMap(countryCodes, countryNames);

String code;

while (true) {

code = stringInput("Enter a code to search for the country. Press Enter to quit: ");

if ((code.length() == 0)) { // if the user presses enter, the code will be an empty string

break; // if nothing is entered, end the loop

}

String countryName = searchCountry(code, countriesAndCodes);

System.out.println(countryName);

}

}

public static Map<String, String> createCountryCodesMap(String[] countryCodes, String[] countryNames) {

Map<String, String> countryCodesMap = new HashMap<>();

for (int i = 0; i < countryCodes.length; i++) {

countryCodesMap.put(countryCodes[i], countryNames[i]);

}

return countryCodesMap;

}

public static String searchCountry(String countryCode, String[] countryCodes, String[] countries) {

for (int i = 0; i < countryCodes.length; i++) {

if (countryCode.equals(countryCodes[i])) {

return countries[i];

}

}

return "Code not found";

}

}

**Q4.**

package week\_5;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import static input.InputUtils.\*;

/\*\*

\* This program contains an example set of data about individual campsites for one day at a campground.

\* People who want to stay at a campsite can make reservations.

\* This program will search the campsite data and return a list of sites that match a user's search criteria,

\* and are available.

\*

\* In this program, the user will enter information about the type of campsite they want, and

\* your program will search for matching campsites, and return a list of campsites that are available.

\*

\* Each campsite has a unique number

\* Each campsite has a type ("RV" or "TENT")

\* Campsites may or may not have water at the site, stored under a "has\_water" key with a value of "YES" or "NO"

\* Campsites can be reserved, stored in a reserved key with a value of "RESERVED" or "AVAILABLE"

\*

\* Finish the getMatchingSites method.

\* Search the siteInfo Map (the global variable) and return a List of campsites that match

\* the search criteria AND are available.

\*

\* - For example, if the user wants an RV campsite and wants water, then campsite 1 and 2 and 8 and 10 match,

\* but only 2 and 8 and 10 are available. Return a list of [2, 8, 10]

\*

\* - For example, if the user wants a TENT campsite and does not want water, then campsites 4, 5, 6 match,

\* but only 4 and 5 are available. Return a list [4, 5]

\*

\* - For example, if the user wants a TENT campsite and does want water, then site 3 matches,

\* but it is reserved. Return an empty list []

\*/

public class Question\_4\_Camping\_Reservations {

/\* Example individual campsite data. Do not modify this.

The keys are:

"type" the intended purpose of this type of site, for recreational vehicles (RV) or tents (TENT).

"has\_water" whether a water connection/faucet is provided at the site or not, "YES" or "NO".

"not\_allowed" A tent is usually permitted at an RV site, and RVs can use tent sites, unless specified here. For example, site 4 only allows tents, no RVs.

"all\_year" Some sites are only open during the summer (SUMMER\_ONLY), others are open all year (AVAILABLE).

"availability" whether the site is currently available, or is reserved

You will need to work with the type, has\_water and availability data.

You will not need to use, and can ignore, the not\_allowed and all\_year data for this program.

\*/

private static Map<String, String> site1 = Map.of("type", "RV", "has\_water", "YES", "not\_allowed", "TENT", "availability", "RESERVED", "all\_year", "AVAILABLE");

private static Map<String, String> site2 = Map.of("type", "RV", "has\_water", "YES", "not\_allowed", "TENT", "availability", "AVAILABLE", "all\_year", "AVAILABLE");

private static Map<String, String> site3 = Map.of("type", "TENT", "has\_water", "YES", "not\_allowed", "RV", "availability", "RESERVED", "all\_year", "AVAILABLE");

private static Map<String, String> site4 = Map.of("type", "TENT", "has\_water", "NO", "not\_allowed", "RV", "availability", "AVAILABLE", "all\_year", "AVAILABLE");

private static Map<String, String> site5 = Map.of("type", "TENT", "has\_water", "NO", "not\_allowed", "RV", "availability", "AVAILABLE", "all\_year", "AVAILABLE");

private static Map<String, String> site6 = Map.of("type", "TENT", "has\_water", "NO", "not\_allowed", "RV", "availability", "RESERVED", "all\_year", "SUMMER\_ONLY");

private static Map<String, String> site7 = Map.of("type", "RV", "has\_water", "NO", "not\_allowed", "TENT", "availability", "AVAILABLE", "all\_year", "SUMMER\_ONLY");

private static Map<String, String> site8 = Map.of("type", "RV", "has\_water", "YES", "not\_allowed", "TENT", "availability", "AVAILABLE", "all\_year", "SUMMER\_ONLY");

private static Map<String, String> site9 = Map.of("type", "RV", "has\_water", "NO", "not\_allowed", "TENT", "availability", "RESERVED", "all\_year", "SUMMER\_ONLY");

private static Map<String, String> site10 = Map.of("type", "RV", "has\_water", "YES", "not\_allowed", "TENT", "availability", "AVAILABLE", "all\_year", "SUMMER\_ONLY");

// Map of all the site reservation data. Do not modify this.

// Use this Map in your getMatchingSites method.

// Notice that the site names are String values, not integers.

static Map<String, Map<String, String>> siteInfo = Map.of("1", site1, "2", site2, "3", site3, "4", site4, "5", site5, "6", site6, "7", site7, "8", site8, "9", site9, "10", site10);

public static void main(String[] args) {

// You do not need to modify the main method.

List<String> siteTypes = List.of("RV", "TENT");

String siteType;

do {

siteType = stringInput("Do you want to reserve an RV or TENT site? Enter RV or TENT.").toUpperCase();

} while (!siteTypes.contains(siteType));

boolean needWater = yesNoInput("Do you want water at the site?");

List<String> availableSiteNames = getMatchingSites(siteType, needWater);

if (availableSiteNames.isEmpty()) {

System.out.println("No sites match your search");

} else {

System.out.println("These sites are available: " + availableSiteNames);

}

}

public static List<Integer> getMatchingSites(String type, boolean hasWater) {

List<Integer> matchingSites = new ArrayList<>();

for (int siteNumber = 1; siteNumber <= 10; siteNumber++) {

Map<String, String> siteInfo = switch (siteNumber) {

case 1 -> site1;

case 2 -> site2;

case 3 -> site3;

case 4 -> site4;

case 5 -> site5;

case 6 -> site6;

case 7 -> site7;

case 8 -> site8;

case 9 -> site9;

case 10 -> site10;

default -> null;

};

if (siteInfo != null && siteInfo.get("type").equals(type) && siteInfo.get("has\_water").equals(hasWater ? "YES" : "NO") && siteInfo.get("availability").equals("AVAILABLE")) {

matchingSites.add(siteNumber);

}

}

return matchingSites;

}

}